Task Assignment Strategies for a Complex Real-time Network System

Hongryeol Kim, Jaejoon Oh, and Daewon Kim

Abstract: In this paper, a study on task assignment strategies for a complex real-time network system is presented. Firstly, two task assignment strategies are proposed to improve previous strategies. The proposed strategies assign tasks with meeting end-to-end real-time constraints, and also with optimizing system utilization through period modulation of the tasks. Consequently, the strategies aim at the optimizationto optimize of system performance with while still meeting real-time constraints. The proposed task assignment strategies are devised using the genetic algorithmswith heuristic real-time constraints in the generation of new populations. The strategies are differentiated by the optimization method of the two objectives-meeting end-to-end real-time constraints and optimizing system utilization: the first one has sequential genetic algorithm routines for the objectives, and the second one has one multiple objective genetic algorithm routine to find a Pareto solution. Secondly, the performances of the proposed strategies and a well-known existing task assignment strategy using the BnB(Branch and Bound) optimization are compared with one other through some simulation tests. Through the comparison of the simulation results, the most adequate task assignment strategies are proposed for someas system requirements-: the optimization of system utilization, the maximization of running tasktasks, and the minimization of the number of network node nodesnumber for a network system.

Keywords: End-to-end, genetic algorithm, real-time, system utilization, task assignment.

1. INTRODUCTION

Various studies have been performed on network systems, since network systems enable the conjunction of various heterogeneous systems and while minimizing the cost for the conjunction simultaneously. These days, these kinds of studies are extended to the field of complex real-time systems, such as autonomous robots and rolling stocks. The complex real-time systems require functional completions in time as well as functional executions with accuracy. However, delays are inevitable in the network system because network nodes in the systems share the network as media for information exchange with others. Network arbitration for media access causes the delays. Common complex real-time systems require several execution flows of their application programs. Multi-tasking operating systems are usual solutions for the system requirement, and delays are also incurred while the application programs are scheduled by the operating systems on the nodes of the network systems. The delays by the operating system scheduling combined with the delays by the network arbitration sometimes generate unexpected long end-to-end delay [1].

For the reason mentioned above, a task assignment strategy that assigns application programs-tasks on network nodes is essential for the design of the real-time network system to guarantee a priori that all deadlines will be met. In addition to the real-time feature, the task assignment strategy can enable the optimization of system utilization, and consequently achieve the optimization of the system performance.

In this paper, a study on the task assignment strategies is presented for a complex real-time network system. The target system of this paper consists of network nodes with real-time multi-tasking operating systems, and the nodes guarantee the real-time deadlines of tasks through the operating systems. The network protocol of the target system is the CAN (Controller Area Network) [2].

Two kinds of real-time scheduling strategies are needed for the target network system. The first one is the real-time scheduling of the operating system. There has been a plethora of studies in the field of real-time scheduling of operating systems, and most of them guarantee real-time scheduling by periodically invocating tasks [3,4]. Undoubtedly, it is much easier to guarantee a priori that all deadlines

Manuscript received April 20, 2005; revised February 6, 2006; accepted May 9, 2006. Recommended by Editor Keum-Shik Hong.

Hongryeol Kim, Jaejoon Oh, and Daewon Kim are with the School of Information Engineering, Myongji University, San 38-2 Nam-dong, Cheoin-gu, Yongin-si, Kyongki-do 449-728, Korea (e-mails: hr.kim@carrier.co.kr, eng-hon@mju.ac.kr, dwkim@mju.ac.kr).

will be met with periodically invocating tasks, rather than with aperiodically invocating tasks. To guarantee a priori that all deadlines will be met, periodical resource assignment was performed even for the aperiodically invocating tasks, as shown in [5]. Real-time scheduling of the operating system in this paper is a well-known static scheduling method-the RMS (Rate Monotonic Scheduling) for periodically invocating tasks. The second one is the real-time scheduling for the network arbitration. There also have also been numerous studies in the field of the network message scheduling. Some good examples of the message scheduling were proposed in the field of the FieldBus. There are two categories for the scheduling: the first one is scheduling by a network master on a network system as proposed in the ProfiBus and the second one is scheduling by every node on a network system and arbitration mechanism of the network, as proposed in the CAN. Particularly for the CAN, many studies have adopted the concept of the operating system scheduling methods [6-8]. Real-time scheduling of the network message in this paper is performed through the encoding method of the CAN message arbitration field with the RMS priority-the release period of the message.

In this paper, two task assignment strategies are introduced to improve upon previous strategies. The proposed strategies assign tasks by meeting end-to-end real-time constraints, and also with optimizing system utilization through period modulation of the tasks. Consequently, the strategies aim at the optimization of system performance while still meeting real-time constraints. The proposed task assignment strategies are devised using the genetic algorithm with heuristic real-time constraints in the generation of new populations. The strategies are differentiated by the optimization method of the two objectives-meeting end-to-end real-time constraints and optimizing system utilization: the first one has sequential genetic algorithm routines for the objectives and the second one has one multiple objective genetic algorithm routine to find a Pareto solution.

In addition to the proposals, the performances of the proposed strategies and a well-known existing task assignment strategy using the BnB (Branch and Bound) optimization are compared with one other through some simulation tests. Through the comparison of the simulation results, the most adequate task assignment strategies are proposed for some system requirements: the optimization of system utilization, the maximization of running tasks, and the minimization of the number of network nodes for a network system.

The technical backgrounds and previous studies are reviewed in Section 2 of this paper, and problem statements of general network systems and the target system of this paper are described in Section 3. A performance criterion is also proposed in Section 3 for the proposal of task assignment strategies in Section 4. The existing task assignment strategy using the BnB optimization is described in detail in Section 4, and the proposal of two task assignment strategies based on the genetic algorithm is performed using the problem statements and the performance criterion shown in Section 3. For the performance evaluation of the existing task assignment strategy and the two proposed task assignment strategies, setups of some simulation tests are introduced and the test results are analyzed by comparison of performance in Section 5. Through the comparison results, the most adequate task assignment strategies are also proposed in Section 5 for some system requirements: the optimization of system utilization, the maximization of running tasks, and the minimization of the number of network nodes for a network system. Finally, the conclusions and proposal for further research are shown in Section 6.

2. TECHNICAL BACKGROUNDS

Since the problem of task assignment considering relationships of task interconnections and task executions is known to be NP-hard problem to find optimal solution [9,10], heuristic optimization methods and meta-heuristic optimization methods such as the simulated annealing and the genetic algorithm are usually used for the optimization.

Chu and Lan [11] chose a heuristic search method, CP/MISF (Critical Path/Most Immediate Successors First) and a heuristic optimization method, DF/IHS (Depth First/Implicit Heuristic Search). They attempted to minimize the processor computation load from the viewpoint of system. The processor load was defined with summation of communications among tasks and accumulated time of task executions in the study. A task assignment strategy was proposed in the study using communication constraints and wait-time ratios. Kohler and Steiglitz [12] proposed a task assignment strategy using the BnB (Branch and Bound) optimization method for a task assignment strategy. The task assignment strategy was also proposed for the purpose of minimizing communications among tasks and execution time of the tasks. However, these task assignment strategies cannot guarantee real-time constraints because minimized communication cost or task execution cost does not mean task delays within the real-time deadlines itself. The delay is considered in [13], but the study focused on the minimization of averaged delay. Since the feasibility of real-time system is defined with the worst-case delay, the task assignment strategy proposed in the study is not also adequate for real-time systems.

To investigate real-time systems, Peng *et al.* [14] proposed a real-time task assignment strategy with the BnB optimization. The task assignment strategy was based on periodic tasks, communication constraints among them, and the deadlines of the communications. Since the study focused on the minimization of the communication cost, there was no consideration of delay caused by priorities of tasks on a network node. A good example of an experiment considering both of the delay caused by priorities of tasks on a network node and the network delay was performed using a heuristic optimization in [15]. However, there was no analysis of end-to-end delay among tasks, so the synchronousness among scheduling strategies existing on network system was also not considered for the delay minimization. The focus of the study was minimizing search time of optimal solution. Another feature of the study is that the proposed task assignment strategy has a constraint, which forces a specified task to be assigned to a specified node. This kind of constraint is very usual in actual implementations because a low-level task controlling a specified hardware resource must be assigned on node with the hardware resource.

Tindell *et al.* [16] chose the simulated annealing aimed at a network system with a well-known type of static scheduling, the DMS (Deadline Monotonic Scheduling) for task scheduling, and with a token-based message scheduling system. The synchronousness among the scheduling strategies existing on the network system was also not considered in the study.

There were several studies using another meta-heuristic optimization method, the genetic algorithm. Chung and Dietz [17] proposed a task assignment strategy using the genetic algorithm that focused on the scheduling of an operating system on a standalone system. Nossal and Galla [18] proposed a task assignment strategy also using the genetic algorithm for real-time the network systems. The task assignment in the study used the genetic algorithm combined with a heuristic search. Because the study also aimed to minimize the communication cost, there was no consideration of delay caused by the priorities of tasks on network nodes. Task assignment strategies using the genetic algorithm can be implemented to be applicable in run time [19,20].

The task assignment strategies have recently become the focus of research with the assumption of a specified application, such as robot systems [21] and process controls [22], or with the assumption of a specified platform, such as a specified operating system scheduling method and a specified network protocol, as shown in [16]. However task modeling and platform resource modeling are still performed with an abstractive model, so the general applicability of the studies can consequently be obtained with the specified applications or even with the specified platforms.

In this paper, task assignment strategies are evaluated for use in a complex real-time network system, so analytical modeling of the worst-case end-to-end delay is presented first. In the worst-case delay modeling, delay caused by the asynchronousness among task scheduling strategies and network message scheduling strategy in the path of task communication is considered, and the system platform for the worst-case modeling is also specified. The system platform consists of an operating system scheduling strategy, network message scheduling, and message manager for the periodic release of the messages and for transparent communication among tasks. Task modeling and platform resource modeling of the system are performed with an abstractive time model for general applicability.

Two task strategies proposed in this paper are based on a meta-heuristic optimization method, the genetic algorithm with heuristic real-time constraints in the generation of new populations. The strategies are differentiated by the optimization method of the two objectives: meeting end-to-end real-time constraints and optimizing system utilization. To our knowledge, this paper is the first to propose task assignment strategies with the two objectives. The proposed strategies also have constraints, which force specified tasks to be assigned at specified nodes, as shown in [15].

In addition to the proposals, the performances of the proposed strategies and a well-known existing task assignment strategy using the BnB optimization [14] are compared with one other through some simulation tests. Through the comparison of the simulation results, the most adequate task assignment strategies are proposed for some system requirements-the optimization of system utilization, the maximization of running tasks, and the minimization of the number of network nodes for a network system. We believe that the comparison of task assignment strategies from the viewpoint of system requirements and the proposal of the most adequate ones is firstly performed in this study, and the study will be useful when the strategies play roles as a part of development tool of real-time network systems.

3. TARGET SYSTEM AND PROBLEM STATEMENTS

The target system of this paper is shown in Fig. 1. One or more tasks are executed on each node of the system, and the real-time of their execution is guaranteed by a real-time multi-tasking operating system on each node, as shown in Fig. 1. The task operation environment with the multi-tasking operating systems shown in Fig. 1 is very usual with

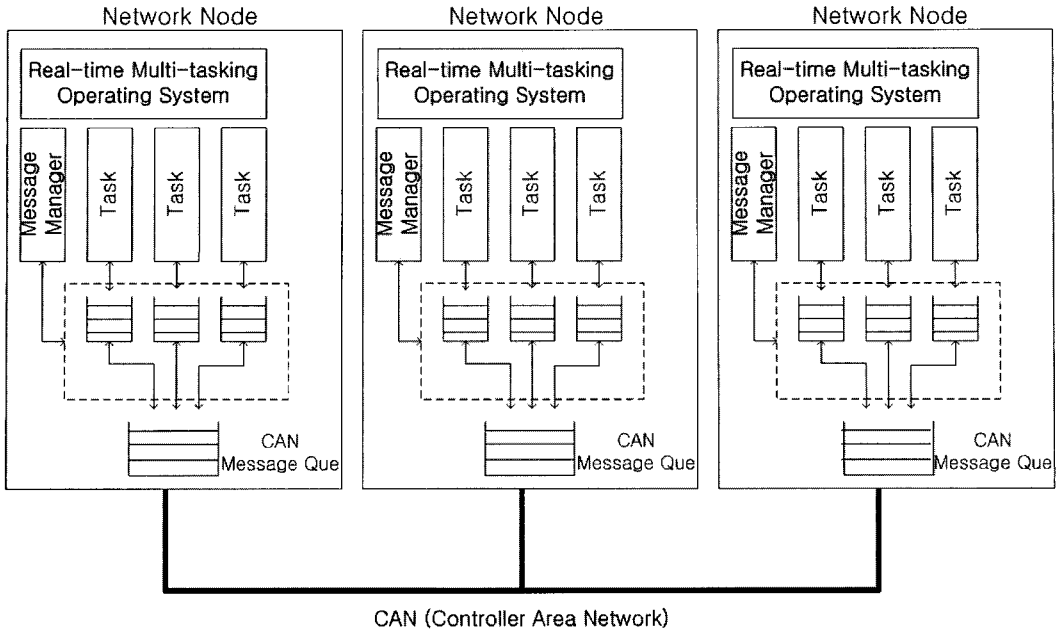


Fig. 1. Network system based on the CAN and multi-task operating system.

complex network systems such as modern network-based autonomous robots. The network nodes in Fig. 1 are interconnected with others through the CAN. A task on a node interfaces with others in the node where the task is assigned or in other nodes through the CAN. The scheduling methods of the multi-tasking operating system are the RMS in this paper, and the scheduling method of the CAN message is implemented by encoding the RMS priority into the arbitration field of the CAN message.

Message transfers among the tasks are performed by the message managers in Fig. 1. The message managers have the highest priority in their own network nodes, and consequently, they can be executed most frequently among the tasks on their own nodes. The message managers transmit messages in the transmitting buffer to the receiving buffers in their own nodes when the receiving tasks of the messages are in the node, or transmit the messages to the transmitting queue of the CAN message when the receiving tasks are in another node. The message managers also retrieve the CAN messages from their receiving queues in the nodes of message managers and transmit the messages to the receiving buffers of receiving tasks of the messages.

Through the service by the message managers, the CAN messages can be released periodically regardless of aperiodic completion time of the tasks, and receiving tasks can be executed asynchronously with their message arrival time [1]. The periodic release of the CAN message is essential to the RMS scheduling of the CAN messages, and the periodic executions of the tasks are also essential to the RMS scheduling of the tasks. Additionally, task can communicate with others transparently regardless of their locations, through the service by the message managers.

A real-time system, particularly a hard real-time system must guarantee a priori that all deadlines will be met. For such a guarantee, analysis of the real-time feasibility is performed based on the analysis of the worst-case delay. In this section, the worst-case delays of tasks and their compositions are analyzed in cases in which the executions of the tasks are dependent on information exchanges among the tasks on the network system. The end-to-end delay is defined as elapse time between the release time of the transmitting task and the completion time of the receiving task through message transmission from the transmitting task to the receiving task. This end-to-end delay is required to be within deadline for the control stability of the task performance.

When a transmitting task and a receiving task share one node, the end-to-end delay is depicted in Fig. 2. In Fig. 2, the transmitting task delay(①) and the receiving task delay(③) are task execution delays caused by scheduling based on the RMS. (1) shows the worst-case execution delay r_i for a task i by the RMS, when blocking time caused by resource sharing can be excluded [3].

$$r_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{r_i}{T_j} \right\rceil C_j, \tag{1}$$

where C_i and C_j are the computation times of the task i and task j individually, T_j is the release period of the task j, and $hp(i)$ is the set of tasks with higher

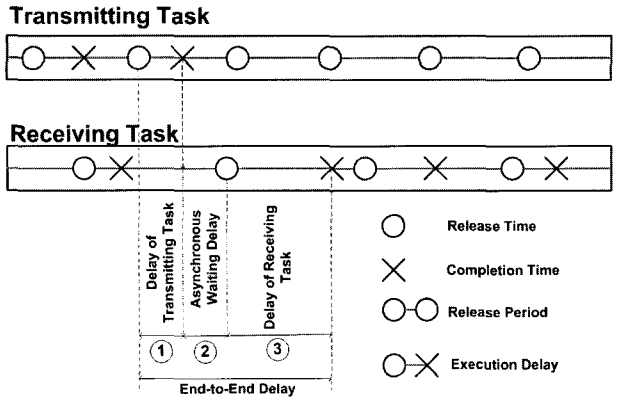


Fig. 2. End-to-end delay on one network node.

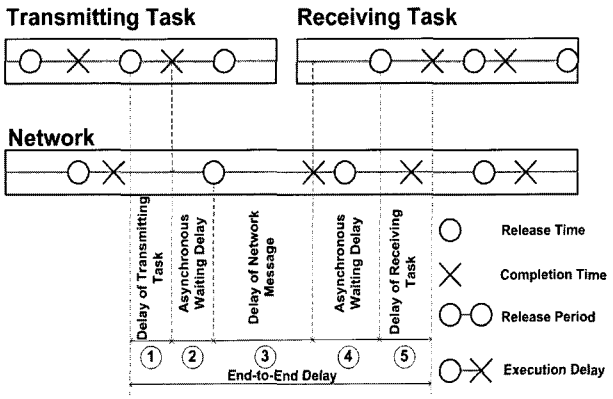


Fig. 3. End-to-end delay between separate network nodes.

priorities than the task i.

In Fig. 2, the transmitting task and the receiving task execute with their own periods. Consequently, waiting time occurs due to the asynchronousness between the completion time of the transmitting task and the release time of the receiving task(②). Since the asynchronous waiting delay cannot be greater than the release period of the receiving task, the worst-case end-to-end delay between tasks sharing one node(r_{ete}) is represented by (2). The delays for the message transfer among message queues and buffers are included in (1), because the transfer is performed by the message manager and the message manager has the highest priority in the node.

$$r_{ete} = r_{trans} + r_{rcv} + T_{rcv}, \tag{2}$$

where r_{trans} and r_{rcv} are the worst-case execution delay(r_i) for the transmitting task and receiving task individually, represented by (1). T_{rcv} is the release period of the receiving task.

When a transmitting task and a receiving task are located at separate network nodes, the end-to-end delay is depicted in Fig. 3. In this case, the tasks exchange information by network message through the CAN. In Fig. 3, The end-to-end delay consists of transmitting task delay(①), receiving task delay(⑤), waiting time caused by the asynchronousness(②,④), and network message delay(③). The transmitting task delay and the receiving task delay are task execution delays caused by scheduling based on the RMS, and are also represented by (1). The waiting time occurs two times in this case. The first waiting time(②) occurs due to the asynchronousness between the completion time of the transmitting task and the release time of the network message. The second waiting time(④) occurs due to the asynchronousness between reception time of the network message and the release time of the receiving task.

The worst-case network message delay r_m of message m is shown in (3) when the arbitration field of the CAN is encoded with the RMS priority [6]. In (3), the transmission delay of the message m is represented by (4). The transmission delay represented by (4) is based on the standard CAN message frame with 11-bit arbitration field. The RMS priority can be encoded into the standard arbitration field as shown in [6].

$$r_m = C_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{r_m + J_j}{T_j} \right\rceil C_j, \tag{3}$$

where C_m and C_j are transmission delays of message m and j individually, T_j is release period of message j, and J_j is the transmission jitter of the message j. The transmission jitter is the delay taking into account the difference in the arbitration start time at different nodes due to propagation delays.

$$C_m = \left(\left\lfloor \frac{34 + 8s_m}{5} \right\rfloor + 47 + 8S_m \right) \tau_{bit}, \tag{4}$$

where S_m is the message size in bytes, τ_{bit} is the time consumed to transmit a bit signal on the network.

Since the first asynchronous waiting delay cannot be greater than the release period of the network message and the second asynchronous waiting delay cannot be greater than the release period of the receiving task, the worst-case end-to-end delay between tasks in separate network node(r_{ete}) is represented by (5). The delays for the message transfer between message queues and buffers are also included in (1).

$$r_{ete} = r_{trans} + T_{net} + r_m + T_{rcv} + r_{rcv}, \tag{5}$$

where T_{net} is the release period of the network message m.

In (4), the message transmission delay is determined by the message size S_m and the transmission time τ_{bit}. In the case of the CAN, ISO

11898-2(CAN High Speed Physical Layer) and ISO 11898-3(CAN Fault Tolerant Physical Layer) can be applicable alternatively for the physical layer protocol of the CAN. The transmission speeds defined by the norms are 1Mbps and 125Kbps individually. Since the maximum data size of the CAN message frame is 8 bytes long, the transmission delay of a message frame of the CAN is about 1ms, even with the lower 125Kbps. Besides, the software timer of the multi-tasking operating systems, such as the Linux, is triggered by periodic tick interrupt, and the trigger period is usually defined with 10ms or more. Actually, only a few powerful multi-tasking operating systems, such as Linux ported for Alpha by Compaq, can guarantee a 10ms or less trigger period [23]. Despite of the worse real-time capability of the Linux compared to other application specific real-time operating systems, trials to adopt the Linux for real-time systems are widely spread because of its familiarity and economical efficiency. In many cases, real-time controls with a few of milliseconds are implemented in standalone controllers, and the Linux is used for higher-level controller integrating the standalone controllers with a few tens of milliseconds or with a few hundreds of milliseconds.

The trigger period is the base unit of the computation time and the period of a task, so both the computation time and the period of the task are represented by the multiples of the trigger period. As mentioned above, the message manager has the shortest period for the minimization of the delay between the completion time of the transmitting task and arrival time of message by the transmitting task at the buffer of the receiving task. Since the transmission delay of a message is small enough compared to the period of the message manager, the release period of all messages are coincide with the period of the message manager. The period of the message manager is configured to guarantee enough bandwidth of the network for all messages of the network system.

Since the summation of the T_{net} and the r_m is comparatively smaller than others, (5) can be rewritten with (6). As shown in (6), the most important factors governing the end-to-end delay based on the RMS are the periods of the transmitting task and receiving task and the calculation time of them.

$$r_{e2e} = r_{trans} + T_{rcv} + r_{rcv} + \gamma, \tag{6}$$

where $r_{trans} + T_{rcv} + r_{rcv} >> \gamma$.

Through the comparison of (2) and (6), the end-to-end delay is governed by the priority of the transmitting task, the priority of the receiving task, their computation time, and the asynchronousness between the periods of the task rather than the communication cost of the network. In other words, the periods and computation time of the tasks are the most important factors that determine the end-to-end delay because the periods are the priorities with scheduling based on the RMS. The computation time is assumed to be a constant defined by (7) in the worst-case delay analysis. The computation time C_i of the task i is determined by the computation power CP of the network node where the task is assigned, the memory size MS, operating system OST, and computation quantity TCS required for task completion. When the computation time is given through the features of a network node, the utilization determines periods of tasks on the network node. To be exact, due to the constraint of available computation resources, the utilization of the resource cannot be greater than 1 in any case. Unfortunately, the utilization constraint of the resource will be reduced depending on scheduling method of operating system. The utilization of a node k is represented by (8), and (9) is the sufficient condition of utilization of the RMS to meet the deadlines of tasks on the node [24]. In (9), h is number of task subsets, where the period of any task can be divisible by all smaller periods of other member tasks. The subsets are called harmonic chains.

$$C_i = f(CP, MS, OST, TCS), \tag{7}$$

$$u_k = \sum_{i=1}^{n} \frac{C_i}{T_i} \le 1, \tag{8}$$

$$u_k < h(2^{\frac{1}{h}} - 1). \tag{9}$$

As shown in (8) and (9), the utilization is enhanced when periods of tasks are shorter, and the constraint of the utilization is determined by the number of harmonic chains. Consequently, the enhancement of the utilization can be achieved through the minimization of harmonic chains of the task periods and through the minimization of the task periods while meeting the constraints of the utilization determined by the harmonic chains. The optimal solution of the task periods for the utilization optimization is known to be NP-complete [24]. The utilization is the periodic execution capability of the system functions, and consequently the functional performance of the system depends on the utilization. Additionally, the utilization determines the number of network nodes to expropriate dedicated tasks and the assignable number of tasks with a dedicated number of network nodes. System utilization U from (8) and end-to-end delay D_{ETE} from (2) and (6) for whole the network system with m nodes and l inter-task communications are shown in (10).

$$U = \sum_{k=1}^{m} u_k, \tag{10}$$

$$D_{ETE} = \sum_{k=1}^{l} r_{ete}.$$

Since the system performance is better with shorter end-to-end delay and higher system utilization, a system performance criterion is proposed in (11) using (10). In (11), α and β are weight factors for the system utilization and for the end-to-end delay individually.

$$P = \frac{1+\alpha U}{1+\beta D_{E2E}} \tag{11}$$

4. TASK ASSIGNMENT STRATEGY

In this section, an existing study on task assignment using the BnB optimization is described in detail, and two task assignment strategies using (11) and the genetic algorithm are proposed. The proposed task assignment strategies optimize both of the end-to-end delay and system utilization. The proposed task assignment strategies are for a homogeneous network system. In other words, the computation time C_i of the task i is assumed to be identical regardless of its assignment.

4.1. Task assignment strategy using the BnB optimization

Task assignment strategy using the BnB optimization performs a task assignment that minimizes system hazard and, consequently, minimizes end-to-end delay [14]. The system hazard Θ^{δ} according to system assignment δ is shown in (12), and normalized value $\bar{C}_{iv}$ is determined with (13). In (12) and (13), T is the task set of whole network system, I_{iv} is v_{th} release time of task i, r_{iv} is end-to-end delay of the v_{th} release time, and d_{iv} is the deadline of the v_{th} release time.

$$\Theta^{\delta} = \max_{T_i \in T}\left(\bar{C}_{iv}\right), \tag{12}$$

$$\bar{C}_{iv} = \frac{r_{iv} - I_{iv}}{d_{iv} - I_{iv}}. \tag{13}$$

The purpose of the task assignment strategy using the BnB optimization is to find optimal task assignment δ^* that minimizes system hazard. Optimal task assignment δ^* is the task assignment that satisfies (14) among all task assignments satisfying the real-time constraint of the system hazard, $\Theta^{\delta} \le 1$.

$$\delta^* = \min_{\delta}\left(\Theta^{\delta}\right) \tag{14}$$

The BnB optimization method evaluates the bound value at each vertex of a search tree, and extension from the vertex is performed when the bound value is the same as or higher than previous optimal solution. The number of vertexes with the BnB is smaller than those of other heuristic optimization methods; consequently, the search time of the BnB is shorter than others.

The task assignment using the BnB optimization generates a tree with a number of levels that are the same as the number of tasks. In other words, when there are m tasks to be assigned, the search tree has m levels. Each vertex of the tree means the partial assignment of the tasks or the whole assignment of the tasks. Since the system hazard is the maximum end-to-end delay, the BnB optimization method, which extends the tree only at vertexes equivalent to or better than the previous optimal solution, provides an effective solution for the task assignment search with minimized system hazard.

The limitation of the proposed task assignment using the BnB optimization for the target system of Fig. 1 is that the assignment tries to minimize end-to-end delay by the minimization of network delay. Consequently, tasks are concentrated upon a few network nodes. The concentrative assignment means the dense population of tasks in the network nodes, and the density means the inevitable increase of task periods to meet constraint of the utilization, as shown in (8) and (9). The end-to-end delay will be increased with the increase of the task periods, as shown in (2) and (6), and this was not considered in the previous study. Additionally, the hardware dependency-constraint of some task assignments into specified network nodes-is not also considered in the task assignment strategy. Nevertheless when the periods of the tasks are not variable, or when it is important to find the minimum number of required network nodes for dedicated tasks, the task assignment is efficient.

4.2. Gene-TAS^{S} task assignment strategy

The optimization method using the genetic algorithm is expected to have a shorter search time compared to ordinary heuristic search methods and to be adequate for the online optimization through its incremental optimization feature. Besides, the optimized result through the genetic algorithm cannot be validated to be the most optimized, so the genetic algorithm is usually adequate for the searches for permissible solutions. In this paper, task assignment strategies using the genetic algorithm and some heuristic rules that exclude generations of new populations are proposed considering further online application of the strategies.

The first task assignment strategy proposed in this paper is the Gene-TAS^{S} (Sequential Task Assignment Strategy based on the Genetic Algorithm). The Gene-

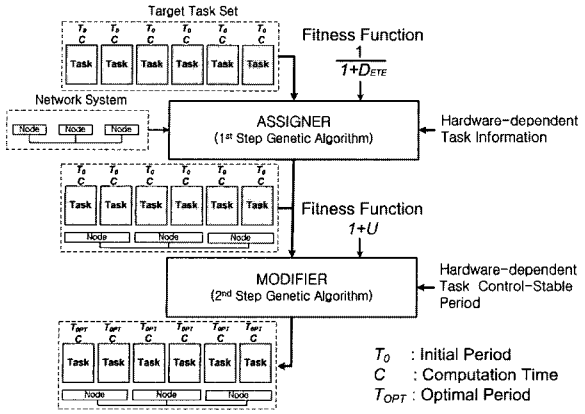


Fig. 4. Gene-TAS^S task assignment strategy.

TAS^S searches optimized task assignments through two sequential steps, as described in Fig. 4. In the first step, in order to guarantee that the end-to-end delays will be within the deadlines, the assigner assigns tasks to network nodes by differentiating hardware-dependent tasks and hardware-independent tasks. In the second step, the modifier modulates the periods of the tasks at the assigned network nodes to higher utilization of the network nodes and, consequently, to higher whole system utilization.

In Fig. 4, the assigner searches permissible task assignments and the modifier optimizes the periods of the tasks through the sequential genetic algorithm routines with a fitness function defined by the system performance criterion P of (11). To be exact, the fitness function of the genetic algorithm for the assigner is the system performance criterion P with $\alpha=0$, $\beta=1$, and the fitness function of the genetic algorithm for the modifier is the system performance criterion P with $\alpha=1, \beta=0$.

The features of the Gene-TAS^S are as follows:

Firstly, the strategy aims to enhance the system performance by the modulations of the task periods as well as the real-time guarantee of end-to-end delay by the assignment of the tasks. Since chromosomes that have possibility of real-time violation are not accepted as new populations with heuristic rules at the modulation step, the modulated periods do not violate real-time constraints.

Secondly, the strategy only finds a permissible task assignment that is permissible within given deadlines, but does not optimize it. Even with the feature, the best task assignment during its search is chosen to be the assignment. Through the search of the just permissible assignment, the time for the search is shorter than time for the optimization.

$$4 \le \frac{T_i^r}{T_i} \le 10, \tag{15}$$

where T_i^r is the rising time of the task performance.

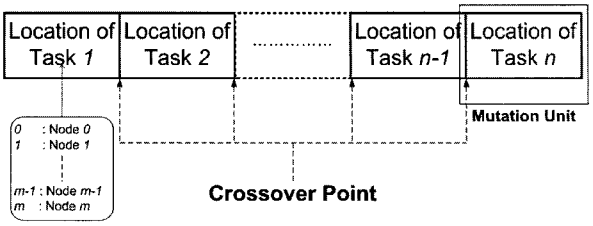


Fig. 5. The chromosome of Gene-TAS^S assigner.

Thirdly, the strategy differentiates hardware-dependent tasks and hardware-independent tasks. Precisely, the assigner assigns the tasks considering that specified tasks have constraints to be located at specified network nodes. This kind of assignment constraint is introduced in [15], and in this paper, the constraints are implemented with a heuristic rule that excludes the generation of new populations. In addition to the constraints of the task assignment, the modulations of the task periods are also constrained with the hardware-dependent tasks for their control stability. In this paper, the constraint is defined with (15), the permissible sampling range of a discrete system [25].

The chromosome of the genetic algorithm for the assigner is shown in Fig. 5. The chromosome consists of fields indicating network nodes assigned for given tasks. Iterative generations of new populations are performed until the convergence of the fitness value. The number of chromosomes of a population is 100. When the difference between the highest fitness value in the last population and the highest fitness value in the present population is less than 0.1%, the assigner judges that the fitness value is converged in the present population. Parents are selected from the last population through the roulette wheel method with probabilities proportional to the fitness values of the chromosomes. With the parents, three genetic operators-crossover, mutation, and elite selection-are used for the new generation of the next population. The rates of the operator invocation are 0.3 for the crossover and 0.5 for the mutation.

The chromosome of the genetic algorithm for the modifier is shown in Fig. 6. The chromosome consists of super fields indicating network nodes, and a network node super filed consists of assigned task periods as its subfield. Iterative generations of new populations are performed until the maximum number of the generations is reached. The number of chromosomes of a population is 100, and the maximum number of the generations is 2,000. Parents are selected form the last population through the roulette wheel method with probabilities proportional to the fitness values of the chromosomes. With the parents, three genetic operators are also used for new generation of the next population. The rates of the operator invocation are 0.3 for the crossover and 0.45

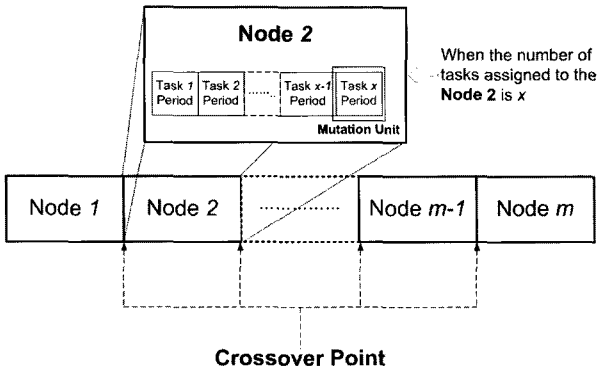


Fig. 6. The chromosome of Gene-TAS^S modifier.

for the mutation. The crossover and mutation are performed with the super field, the field of the network nodes to preserve assigned location of the tasks.

Heuristic rules that exclude the generation of new chromosomes are applied to both of the assigner and the modifier. The heuristic rules as follows:

1) All the worst-case delays of task communications are calculated with (2) and (6) for new chromosomes when the next population is generated. The delays must be within the deadline, and any chromosome violating this rule is excluded in the population.
2) The new chromosome must satisfy (9) at every network node. Any chromosome violating this rule is excluded in the population.
3) The new chromosome must satisfy the hardware dependency of the hardware-dependent tasks. Any chromosome violating this rule is excluded in the population.
4) The new chromosome must satisfy the period constraints of the hardware-dependent tasks defined with (15). Any chromosome violating this rule is excluded in the population.

Although the Gene-TAS^S has the two objectives-the real-time guarantee and the optimization of the system utilization-, finding one optimal solution for both of the objectives cannot be achieved. The shortcoming of the Gene-TAS^S is that it has no way to guarantee real-time constraints through the modulation of the task periods because the real-time guarantee is achieved by the assigner prior to the modifier. Consequently, the search space of the Gene-TAS^S cannot cover all feasible solutions.

4.3. Gene-TAS^P task assignment strategy

As mentioned in Section 3, the problem of the task assignment strategy proposed in this paper is a kind of multiple objective optimization problem. Generally, a solution that is the best adapted to all objectives at the same time does not exist. There are methods, such as the weighted sum method, constraint method, and goal programming method, to solve the multiple objective optimization problems [26]. The problem with these methods is that a priori quantitative evaluation is required for the methods, and the evaluation must conform to the objectives of the optimization. The genetic algorithm is known to be adequate for the multiple objective problem because the genetic algorithm can find several Pareto solutions in its iteratively generated populations [27].

The multiple objective genetic algorithm has multiple fitness functions and collects the Pareto solutions through the weighted combination of the fitness functions. The Pareto ranking method, the Pareto tournament method, and the Pareto reservation method are well-known methods among them.

The second task assignment strategy proposed in this paper is the Gene-TAS^P (Parallel Task Assignment Strategy based on the Genetic algorithm). The Gene-TAS^P searches optimized task assignment through the concurrent execution of the task assignment and the period modulation, as described in Fig. 7. The Gene-TAS^P also has a system performance criterion P of (11) as its fitness function. The vector combination of the fitness function for the Pareto solution is performed through $\alpha=2$, $\beta=1$ for the Gene-TAS^P because feasible solutions of the real-time constraints are permissible.

The chromosome of the Gene-TAS^P is shown in Fig 8. The chromosome consists of super fields indicating given tasks, and a task super field consists of its

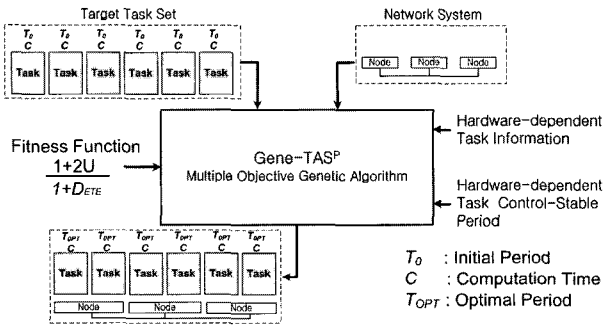


Fig. 7. Gene-TAS^P task assignment strategy.

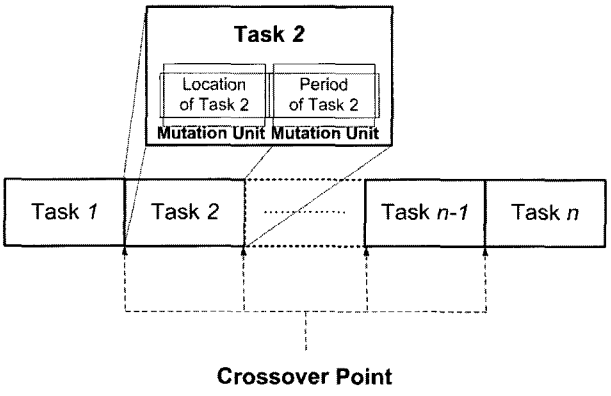


Fig. 8. The chromosome of the Gene-TAS^P.

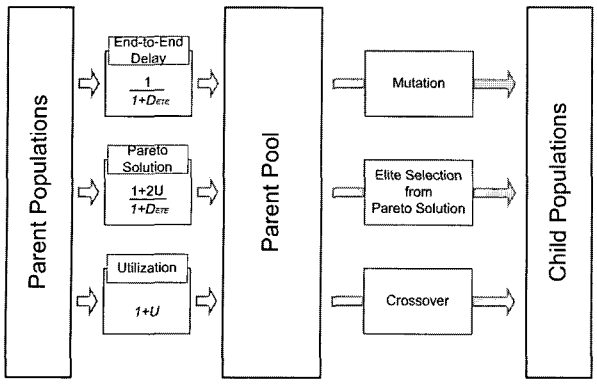


Fig. 9. New generation of the Gene-TAS^P.

location and its period. Iterative generations of new populations are performed until the maximum number of the generations is reached. The number of chromosomes of a population is 100, and the maximum number of the generations is 50,000. As shown in Fig. 9, there is a parent pool for the reproduction. Through the three fitness functions of the Gene-TAS^P, the parents having the optimal end-to-end delay, the parents having the optimal system utilization, and the parents having the optimal system criterion value are selected as the parents with equal numbers for the reproduction. The parent selection method is similar to the method of the Pareto reservation method [28]. Selections of the parents having the optimal end-to-end delay and the parents having the optimal system utilization are similar to the VEGA selection method, where parents having optimal performance with one objective are selected as parents for the reproduction.

With the parents, three genetic operators-crossover, mutation, and elite selection-are used for the new generation of the next population. The rates of the invocation are 0.3 for the crossover and 0.5 for the mutation. Here, the crossover is performed with the super field, but the mutation is performed on the subfields. The selection of a subfield for the mutation between the field of location and the field of period is performed randomly. The heuristic rules that exclude the generation of new chromosomes for the Gene-TAS^S are also applied to the Gene-TAS^P.

5. SIMULATIONS

In this section, some simulation tests are performed with the assumption of m network nodes, n tasks, and k hardware-dependent tasks. Configurations of the tasks are shown in Table 1. The 'H/W' of Table 1 indicates whether the task depends on hardware resource of a network node or not. The number means the number of network node on which the task depends. The initial period T_i of the hardware-dependent tasks are configured as $T_i^r/4$ in Table 1.

Table 1. Task configurations for simulations.

No.	T (ms)	C (ms)	H/W	PRED	Data (Byte)	Deadline (ms)
1	50	20	None	None	None	200
2	50	20	1	1	4	200
3	80	30	2	2	2	260
4	100	40	3	3	8	360
5	100	40	None	4	5	400
6	50	20	None	3	4	260
7	80	30	None	3	8	400
8	50	20	None	5	4	400
9	100	40	None	8	8	300
10	50	20	None	5	6	350
11	100	40	None	5	5	400
12	100	40	None	5	5	400

The 'PRED' of Table 1 transmits messages to the indicated tasks. The number means the number of predecessor from which the task receives messages. For examples, task number 1 transmits messages to task number 2, and task number 2 receives messages from task number 1 and transmits messages to task number 3, as shown in the Table 1. The data size is the size of the data field of the CAN messages from the predecessors. Simulations are performed with three task assignment strategies described in Section 4, including two proposed task assignment strategies.

The purposes and methods of the simulations are described as follows:

Firstly, to find the task assignment that can guarantee the greatest system utilization, the system utilization of the three task assignment strategies with a constant number of tasks and an increasing number of network nodes is analyzed. An analysis of the system utilization is also performed similarly with a constant number of network nodes and an increasing number of tasks. Through the analysis, the task assignment strategy that can guarantee the greatest system performance is proposed.

Secondly, to find the task assignment strategy that can expropriate tasks with the highest numbers, three task assignments are simulated with a constant number of network nodes and an increasing number of tasks with real-time constraints. Through the simulation, the task assignment strategy that can guarantee the greatest amount of system flexibility and system extendibility is proposed.

Thirdly, to find the task assignment strategy that can assign a constant number of tasks with the lowest number of network nodes, the task assignments are simulated with the constant number of tasks. Through the simulations, the task assignment strategy that can minimize the system requirement the most is proposed.

Figs. 10 and 11 are individual system utilizations and the worst-case delays, which are nearest to their

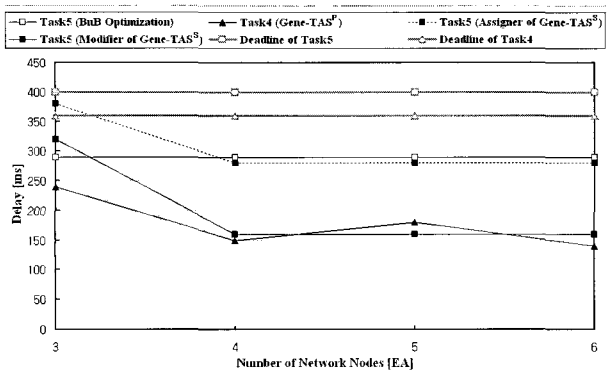


Fig. 10. Delays nearest to the deadlines with an increased number of network nodes.

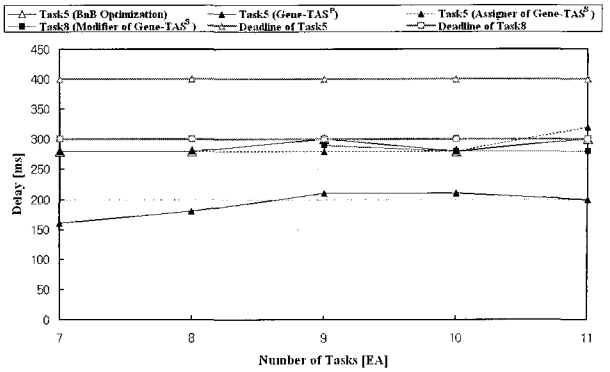


Fig. 12. Delays nearest to the deadlines with the increased number of tasks.

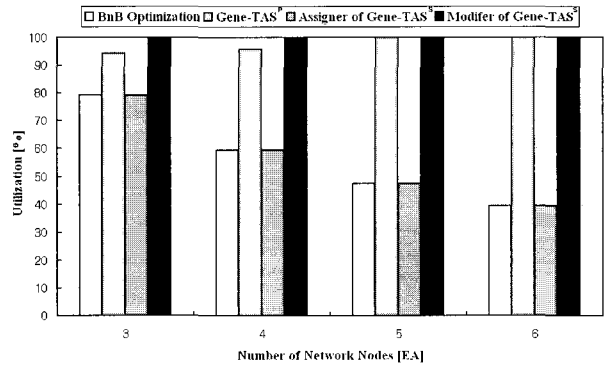


Fig. 11. System utilization with an increased number of network nodes.

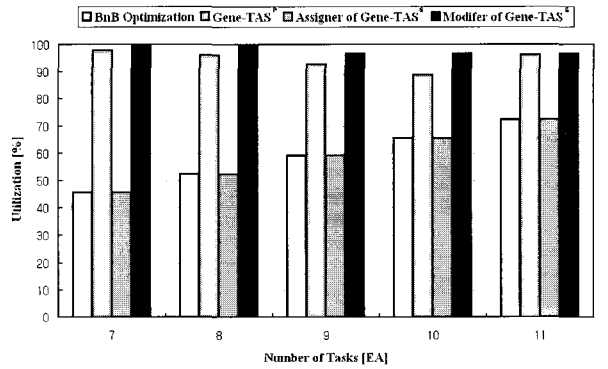


Fig. 13. System utilization with the increased number of tasks.

given deadlines, with six tasks and an increased number of network nodes from three to six. Even with the increased number of network nodes, all task assignment strategies meet the deadline constraints in Fig. 10. and In Fig. 11, the system utilization is decreased with the increased number of network nodes in the cases of the task assignment using the BnB optimization and the assigner of the Gene-TAS^S. The decrease of the utilization occurs because the minimization of the delay is only the purpose of the strategy with the BnB optimization and the assigner of the Gene-TAS^S. In particular, the number of network nodes with assigned tasks is converged with the two strategies because they try to assign the tasks without network communication. Consequently, idle network nodes without assigned tasks decrease the whole system utilization.

On the other hand, the modifier of the Gene-TAS^S and the Gene-TAS^P modulate the periods of the tasks, so the system utilization is maximized, as shown in Fig. 11. The system utilization by the modifier of the Gene-TAS^S is higher than the utilization of the Gene-TAS^P because the modifier modulates the period in the last step of the Gene-TAS^P without any consideration, but the Gene-TAS^P searches for the Pareto solutions, not the sole solution of the maximum system utilization.

Figs. 12 and 13 are individual system utilizations and worst-case delays, which are nearest to their given deadlines, with six network nodes and an increased number of tasks from seven to eleven. Even with the increased number of tasks, all task assignment strategies meet the deadline constraints in Fig. 12, Contrary to the cases of Fig. 10. In Fig. 13, the system utilization is increased with the increased number of tasks, in the cases of the task assignment using the BnB optimization and the assigner of the Gene-TAS^S. The increase of the utilization is due to the increase of the task number with a constant number of network nodes.

However, when the modifier of the Gene-TAS^S and the Gene-TAS^P modulate the periods of the tasks, the system utilization is maximized, as shown in Fig. 13 similar to Fig. 11. The system utilization by the modifier of the Gene-TAS^S is also higher than the utilization of the Gene-TAS^P for the same reason as in Fig. 11.

Through the analysis of the end-to-end delay and system utilization of the three task assignment strategies, the final assignment result of the Gene-TAS^S assigns tasks with the greatest system utilization regardless of the number of network nodes and the number of tasks, when the number of tasks can be assigned while still meeting deadlines. Consequently,

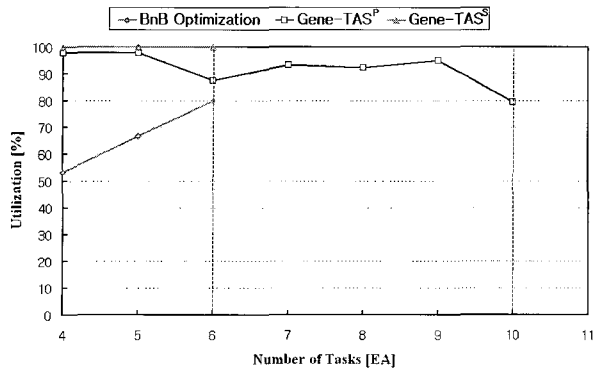


Fig. 14. The permissible number of tasks with three network nodes.

the Gene-TAS^S is the best choice when system requirement is the maximum system performance and the resource of the system is enough to assign dedicated tasks.

The permissible numbers of tasks that can be assigned on three network nodes by the task assignment strategies are shown in Fig. 14. The permissible numbers by the strategy with the BnB optimization and by the Gene-TAS^S are six but the permissible number by the Gene-TAS^P is ten. The result of the Gene-TAS^P is achieved through searching for the Pareto solutions of end-to-end delay and system utilization. Consequently, the search area of the Gene-TAS^P covers a much wider area than others.

As shown in Fig. 14, the Gene-TAS^P can assign the highest number of tasks. Consequently, Gene-TAS^P is the best choice when system requirement is the maximum number of tasks when the number of network nodes is constrained. This capability of the Gene-TAS^P is very important in real applications. Since the predefined number of network nodes is usual system constraint, the Gene-TAS^P enables the system capacity the maximum. This kind of benefit of the Gene-TAS^P is achieved through real-time guarantee through the modulation of the task periods. Despite of the highest utilization of the assignment with the Gene-TAS^S below six tasks, the Gene-TAS^S cannot find any assignment with seven or more tasks because it cannot modulate the task period for the purpose of real-time guarantee.

The numbers of required network nodes with four tasks are shown in the Table 2. As shown in the table, the number of required network nodes is smallest with the task assignment by the strategy with the BnB optimization and by the Gene-TAS^S. Since the system utilization is not the optimization target in this case, the strategy with the BnB optimization and the assigner of the Gene-TAS^S find the task assignment with minimum number of network nodes. These methods attempt to find the task assignment with minimum network communication, but the Gene-TAS^P searches for the Pareto solution considering

Table 2. Required number of network nodes with four tasks.

	Strategy with the BnB	Gene-TASS	Gene-TASP
Required no. of network nodes	2	2	4

system utilization. The Gene-TAS^P aims to increase the number of network nodes when the system utilization can be increased with an increased number of network nodes.

As shown in the simulation results, the Gene-TAS^S is the best choice for a system in which system performance is the most important, and the Gene-TAS^P is the best choice for a system in which a higher number of running tasks is the most important. To assign predetermined number of tasks to minimum network nodes, the Gene-TAS^S and the strategy with the BnB optimization are the best choice.

6. CONCLUSIONS

In this paper, a study on task assignment strategies was performed for the complex real-time network system. Two task assignment strategies, the Gene-TAS^s and the Gene-TAS^p, are proposed to improve upon previous strategies. The proposed task assignment strategies generate task assignments that guarantee end-to-end real-time constraints and also optimize system utilization through rate modulation. Consequently, the strategies aim to optimize the system performance. The proposed task assignment strategies are devised using the genetic algorithm with the heuristic real-time constraints in the generation of new populations. The strategies are differentiated by the optimization method of two objectives, end-to-end real-time guarantees and system utilization. The first one has sequential genetic algorithm routines for the objectives, and the second one has one multiple objective genetic algorithm routine to find a Pareto solution. In addition to the proposals, the performances of the proposed strategies and an existing task assignment strategy using the BnB optimization are compared with one other through some simulation tests. According to the comparison of the simulation results, the Gene-TAS^S is revealed to be the best choice for a system in which system performance is the most important and the Gene-TAS^P is revealed to be the best choice for a system in which the acceptance of large number of tasks is the most important. To assign a predetermined number of tasks to the minimum network nodes, the Gene-TAS^S and the strategy with the BnB optimization are revealed to be the best choices.

For further study, the proposed task assignment strategy will be revised for dynamic assignment of tasks during the run-time of the network system. The

online revision of the task assignment strategy will be useful for flexible online resource management of the system.

REFERENCES

[1] H. Kim, J. Kim, and D. Kim, "Development of coordinated scheduling strategy with end-to-end respose time analysis for the CAN-based distributed control systems," *Proc. of IEEE/RSJ International Conference on Intelligent Robot and System*, pp. 2099-2104, 2004.

[2] *CAN Specification Version 2.0*, Robert Bosch GmbH, 1991.

[3] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," *Proc. of IEEE Real-Time Systems Symposium*, 1989.

[4] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-realtime environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46-61, 1973.

[5] M. Spuri and G. Buttazzo, "Scheduling aperiodic tasks in dynamic priority systems," *Journal of Real-time Systems*, vol. 10, pp. 179-210, 1996.

[6] K. Tindell and A. Burns, "Guaranteed message latencies for distributed safety-critical hard real time control networks," *Technical Report YCS 94-229*, Dept. Computer Science, Univ. of York, York, UK, 1994.

[7] M. Di Natale, "Scheduling the CAN bus with earliest deadline techniques," *Proc. of the IEEE Real-Time Symposium*, pp. 259-268, 2000.

[8] K. M. Zuberi and K. G. Shin, "Non-preemptive scheduling of messages on controller area network for real-time control applications," *Proc. of Real-Time Technology and Applications Symposium*, pp. 240-249, 1995.

[9] J. B. Sinclair, "Efficient computation of optimal assignments for distributed tasks," *Journal of Parallel and Distributed Computing*, vol. 4, pp. 342-362, 1987.

[10] V. M. Lo, "Heuristic Algorithms for task assignment in distributed systems," *IEEE Trans. on Computers*, vol. 37, no. 11, pp. 1384-1397, 1988.

[11] W. W. Chu and L. M. Lan, "Task allocation and precedence relations for distributed real-time system," *IEEE Trans. on Computers*, vol. 36, no. 6, pp. 667-679, 1987.

[12] W. H. Kohler and K. Steiglitz, "Computer and job-shop scheduling theory," *Enumerative and Iterative Computational Approach*, pp. 229-287, John Wiley & Sons, 1976.

[13] W. W. Chu and K. Leung, "Module replication and assignment for real-time distributed processing systems," *Proc. of the IEEE*, vol. 75, no. 5, pp. 547-562, 1987.

[14] D.-T. Peng, K. G. Shin, and T. F. Abdelzaher, "Assignment and scheduling communicating periodic tasks in distributed real-time systems," *IEEE Trans. on Software Engineering*, vol. 23, no. 12, pp. 745-758, 1997.

[15] M. A. Moncusi, J. M. Banus, J. Labarta, and A. Arenas, "A new heuristic algorithm to assign priorities and resources to tasks with end-to-end deadlines," *Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications*, vol. IV, pp. 2102-2108, 2001.

[16] K. Tindell, A. Burns, and A. J. Wellings, "Allocating real-time tasks (an np-hard problem made easy)," *Journal of Real-Time Systems*, vol. 4, no. 2, pp. 145-165, 1992.

[17] T. M. Chung and H. G. Dietz, "Adaptive genetic algorithm: Scheduling hard real-time control programs with arbitrary timing constraints," *Technical Report of Purde University*, 1995.

[18] R. Nossal and T. M. Galla, "Solving NP-complete problems in real-time system design by multichromosome genetic algorithms," *Proc. of SIGPLAN Workshop on Languages, Compilers, and Tools for Real-Time Systems*, pp. 68-76, 1997.

[19] A. S. Wu, H. Yu, S. Jin, K. Lin, and G. Schivone, "An incremental genetic approach to multiprocessor scheduling," *IEEE Trans. on Parallel and Distributed Systems*, vol. 15, no. 9, pp. 824-834, 2004.

[20] Y.-H. Lee and C. Chen, "A modified genetic algorithm for task scheduling in multiprocessor systems," *Proc. of Workshops on Compiler Techniques for High-Performance Computing*, 2003.

[21] T. C. Lueth and T. Laengle, "Task description, decomposition, and allocation in a distributed autonomous multi-agent robot system," *Proc. of International Conference of Intelligent Robots and Systems*, pp. 1516- 1523, 1994.

[22] P. Altenbernd, C. Ditze, P. Laplante, and W. Halang, "Allocation of Periodic real-time tasks," *Proc. of IFAC/IFIP Workshop*, 1995.

[23] "Alpha Linux Ready Systems." Available: http://h18002.www1.hp.com/alphaserver/linux/

[24] T. W. Kuo and A. K. Mok, "Incremental reconfiguration and load adjustment in adaptive real-time systems," *IEEE Trans. on Computers*, vol. 46, no. 12, pp. 1313-1324, 1997.

[25] K. J. Astrom and B. Wittenmark, *Computer-Controlled Systems*, Prentice Hall, 1997.

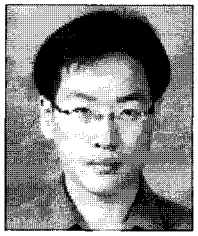
[26] C. L. Hwang and K. Yoon, *Multiple Attribute Decision Making, Methods and Application, a State-of-Art Survey*, Springer-Verlag, 1981.

[27] J.-H. Chen, "Theoretical analysis of multi-objective genetic algorithms - convergence time,

population sizing and disequilibrium," *Report for IEEE NNS Walter Karplus Research Grant*, 2003.

[28] H. Tamaki, H. Kita, and S. Kobayasi, "Multi-objective optimization by genetic algorithms: A Review," *Proc. of the IEEE International Conference on Evolutionary Computation*, pp. 517-522, 1996.

Hongryeol Kim received the B.S. and M.S. degrees in Electrical Engineering from Myongji University in 1996 and 1998. He joined R&D Center of Carrier Korea Ltd. in 1998. He is currently pursuing a Ph.D. degree at the School of Information Engineering, Myongji University. His research interests include real-time network controls and intelligent robot control architectures.



Jaejoon Oh received the B.S. degree in Mechanical Engineering from Kookmin University in 2002, and M.S. degree in Information Engineering from Myongji University in 2004. Currently, he joined R&D Center of the ATI Ltd. His research interests include robot controls and system control architectures.

Daewon Kim received the B.S., M.S., and Ph.D. degrees in Control and Measurement Engineer-ing from Seoul National University in 1983, 1985, and 1990. He was a Senior Engineer at the R&D Center of Daewoo Heavy Industry Ltd. from 1987 to 1992. He is a Professor in the Department of Information Engineering is Myongji University from 1992. His research interests include distributed control systems and personal robots.