Optimization of Task Allocation and Priority Assignment in Hard Real-Time Distributed Systems

QI ZHU, Intel Corp.
HAIBO ZENG, General Motors R&D
WEI ZHENG, University of California, Berkeley
MARCO DI NATALE, Scuola Superiore S. Anna
ALBERTO SANGIOVANNI-VINCENTELLI, University of California, Berkeley

The complexity and physical distribution of modern active safety, chassis, and powertrain automotive applications requires the use of distributed architectures. Complex functions designed as networks of function blocks exchanging signal information are deployed onto the physical HW and implemented in a SW architecture consisting of a set of tasks and messages. The typical configuration features priority-based scheduling of tasks and messages and imposes end-to-end deadlines. In this work, we present and compare formulations and procedures for the optimization of the task allocation, the signal to message mapping, and the assignment of priorities to tasks and messages in order to meet end-to-end deadline constraints and minimize latencies. Our formulations leverage worst-case response time analysis within a mixed integer linear optimization framework and are compared for performance against a simulated annealing implementation. The methods are applied for evaluation to an automotive case study of complexity comparable to industrial design problems.

Categories and Subject Descriptors: C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; G.1.6 [**Optimization**]: Integer programming

General Terms: Design

Additional Key Words and Phrases: Design optimization, real-time systems, schedulability, automotive systems, architectures, optimization

ACM Reference Format:
Zhu, Q., Zeng, H., Zheng, W., Di Natale, M., and Sangiovanni-Vincentelli, A. 2012. Optimization of task allocation and priority assignment in hard real-time distributed systems. ACM Trans. Embedd. Comput. Syst. 11, 4, Article 85 (December 2012), 30 pages.
DOI = 10.1145/2362336.2362352 http://doi.acm.org/10.1145/2362336.2362352

1. INTRODUCTION

Function development in Electronics, Controls, and Software (ECS) vehicle architectures has traditionally been component or subsystem focused. In recent years, there has been a shift from the single Electronic Control Unit (ECU) approach towards an increased networking of control modules with an increased number of distributed time-critical functions and multiple tasks on each ECU.

Q. Zhu is currently affiliated with the University of California, Riverside.
Authors' addresses: Q. Zhu (corresponding author), Department of Electrical Engineering, University of California, Riverside, Riverside, CA 92521; email: qzhu@ee.ucr.edu; H. Zeng, General Motors R&D, 422 Portage Ave., Palo Alto, CA 94306; W. Zheng and A. Sangiovanni-Vincentelli, Department of Electrical Engineering and Computer Sciences, University of California Berkeley, Berkeley, CA 94720; M. Di Natale, Scuola Superiore S. Anna, Via Moruzzi, 1, Pisa, Italy.

DOI 10.1145/2362336.2362352 http://doi.acm.org/10.1145/2362336.2362352

The starting point for the definition of a car electronic/software system is the specification of the set of functions that the system needs to provide. *Functional models* are created from the decomposition of the system-level functions in a hierarchical network of component blocks. The physical architecture model captures the topology of the car network, including the communication buses (e.g., Controller Area Network (CAN)), the ECUs, and the policies that control the shared resources.

The system designer must find a mapping of the functional architecture onto the physical architecture that satisfies the timing requirements (typically sensor-actuator deadlines). The mapping is performed at the time the SW implementation is defined, when the functions are implemented by a set of concurrent tasks and the communication signals are transferred in the payload content of messages.

To provide design-time guarantees on timing constraints, different design and scheduling methodologies can be used. Because of resource efficiency, most automotive controls are designed based on runtime priority-based scheduling of tasks and messages as opposed to time-triggered scheduling [Padmanabhan et al. 1999; Brook 2000; Aström and Wittenmark 1990; Henriksson et al. 2002]. Examples of standards supporting this scheduling model are the OSEK operating system standard [OSEK 2006], AUTOSAR [AUTOSAR 2010a], and the CAN bus [Bosch 1991] arbitration model. Other scheduling policies, including those with a dynamic assignment of priorities such as the Earliest Deadline First (EDF) [Liu and Layland 1973], while quite popular in the research community, are not supported by standards recommendations like OSEK and later AUTOSAR. They have practically no available commercial implementation and offer very limited advantage in terms of schedulability when the tasks to be scheduled have harmonic periods (or at most belong to two or three harmonic sets) as is in almost all automotive control applications.

In the typical model that is used for the implementation of distributed computations, periodic tasks and messages communicate according to a semantics in which the communication channel holds the last value that is written into it and is implemented as a shared variable protected against concurrent access [AUTOSAR 2010b]. This model, called a *periodic activation model*, has some advantages, including the separation of concerns when evaluating the schedulability of the individual resources. It also allows for a very simple specification at the interface of each subsystem or component, thereby simplifying the interaction with the suppliers. The drawback is a nondeterministic time behavior and a possibly large worst-case end-to-end delay in the computations.

The execution model considered in this article is the following. Input data (generated by a sensor, e.g.) are available at one of the system's ECUs. A periodic activation event from a local clock triggers an application task on this ECU. The task reads the input data signal, computes intermediate results as output signals, and writes them to the output buffer from where they can be read by another task or used for assembling the data content of a message. Messages, also periodically activated, transfer the data from the output buffer on the current ECU over the bus to an input buffer on another ECU. Eventually, task outputs are sent to a system output (an actuator, e.g.). The application typically imposes end-to-end latency requirements between a subset of the source-sink task pairs in the system.

1.1. Design Flow

The optimization of the allocation of tasks, the definition of the mapping of signals into messages, and the priority assignment to tasks and messages are the design stages addressed in this work as part of the larger design flow shown in Figure 1. The design flow is based on the Y-chart approach [Kienhuis et al. 2002] where the application description and the architectural description are initially separated and joined together later in a mapping step. In the application model, nodes represent function blocks and

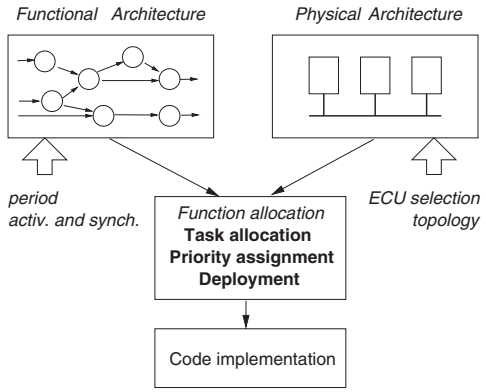

Fig. 1. Design flow stages and the steps (in bold) for which decision variables are optimized according to the timing constraints and the metric objectives.

edges represent data dependencies, which consist of signal information. The application description is further characterized by end-to-end latency constraints along selected paths from sources to sinks. The architectural description is a topology consisting of ECUs connected with buses. In this work, we assume *heterogeneous ECUs*, that is, units with processors of possibly different type and speed, which run a priority-based preemptive OSEK-compliant operating system. Furthermore, we target the case of architectures with possibly multiple *CAN buses*, featuring nonpreemptive priority-based message scheduling. Currently the CAN communication standard represents the large majority of the communication links, with LIN connections used for a relatively small number of local low-speed data communications [Davis et al. 2007].

Mapping deploys functional blocks to tasks (this problem is not handled in this work) and tasks to ECUs. Correspondingly, signals can be mapped into local communication or messages that are exchanged over the buses. Within the mapping step are the operations of *task allocation, signal to message assignment,* and *priority assignment*. We propose an integrated approach in which task allocation, signal packing, and priority assignments are performed in an integrated way. We see an integrated approach as the natural solution for these types of problems given the cross-dependencies among the different aspects of the problem. Messages arise from the definition of task allocation, which is defined (among others) according to the time constraints. The satisfaction of deadline constraints depends on the task allocation and the priority assignment, while assigning priorities requires knowledge of the task allocation and the definition of the message set.

1.2. Prior Work

Both static and dynamic priority, distributed as well as centralized, scheduling methods have been proposed in the past for distributed systems. Static and centralized scheduling is typical of time-triggered design methodologies, like the Time-Triggered Architecture (TTA) [Kopetz et al. 1989] and its network protocol TTP, and of implementations of synchronous reactive models, including Esterel and Lustre [Benveniste et al. 2003]. Also, the recent FlexRay automotive communication standard [Flexray 2006] provides two transmission windows, one dedicated to time-driven periodic streams and the other for asynchronous event-driven communication.

Priority-based scheduling is also very popular in control applications. It is supported by the CAN network arbitration protocol [Bosch 1991]. The response times of real-time CAN messages (with timing constraints) have been analyzed and computed for

the worst case [Davis et al. 2007]. Also, the OSEK operating system standard for automotive applications [OSEK 2006] and the newer AUTOSAR standard [AUTOSAR 2010a] support not only priority scheduling, but also an implementation of the immediate priority ceiling protocol [Sha et al. 1990] for sharing resources with predictable worst-case blocking time. Priority-based scheduling of single processor systems has been thoroughly analyzed with respect to worst-case response time and feasibility conditions [Harbour et al. 1994; Liu 2000].

End-to-end deadlines have been discussed in research work in the context of single-processor as well as distributed architectures. The synthesis of the task parameters (activation rates and offsets) and (partly) of the task configuration itself in order to guarantee end-to-end deadlines in single processor applications is discussed in Gerber et al. [1995]. Later, the work was tentatively extended to distributed systems [Saksena and Hong 1996]. The periodic activation model with asynchronous communication can be analyzed quite easily in the worst case, because it allows the decomposition of the end-to-end schedulability problem in local instances of the problem, one for each resource (ECU or network). In other transaction-based activation models (such as the holistic model in Tindell [1993], the transaction model with offsets in Gutiérrez et al. [1997] and Palencia and Harbour [1998], and the jitter propagation model in Hamann et al. [2004]), messages are queued by sender tasks whenever information is ready and the arrival of messages at the destination node triggers the activation of the receiver task. In such models, task and message schedulers on CPUs and networks have cross dependencies because of the propagation of the activation signals. When systems are constructed according to these models, distributed hard real-time analysis can be performed using the holistic model [Tindell 1993; Pop et al. 2002] based on the propagation of the release jitter along the computation path. When activation offsets can be enforced for tasks and messages to synchronize activations and enforce timing constraints, the analysis is as described by Palencia and Harbour [1999]. However, in most automotive systems with CAN buses, ECUs are not synchronized. This prevents the synchronization of remote tasks and messages activations and the enforcement of a precedence order by the assignment of suitable activation offsets. As opposed to the previously cited transaction models, our activation model is both jitter- and offset-free.

Other methods have been defined for scheduling periodic tasks and messages in a distributed time-triggered architectures. One early proposal can be found in Ramamritham et al. [1993]

While these works provide analysis procedures with reduced pessimism, increasing speed and precision, the synthesis problem is largely open, except for Racu et al. [2005], where the authors discuss the use of genetic algorithms for optimizing priority and period assignments with respect to a number of constraints, including end-to-end deadlines and jitter and an extensibility metric. In Bini et al. [2006], the authors describe a procedure for period assignment on priority-scheduled single-processor systems. In Pop et al. [2003], a design optimization heuristics-based algorithm for mixed time-triggered and event-triggered systems is proposed. The algorithm, however, assumes that nodes are synchronized and the bus transmission time is allocated according to the Universal Communication Model. More recently, an integrated optimization framework is proposed in He et al. [2009] for systems with periodic tasks on a network of processor nodes connected by a bus based on the time-triggered protocol. The framework uses Simulated Annealing (SA) combined with geometric programming to hierarchically explore task allocation, task priority assignment, task period assignment, and bus access configuration. In Davis and Burns [2009] the problem of priority assignment in multiprocessor real-time systems using global fixed-priority preemptive scheduling is addressed. In Lakshmanan et al. [2009] the authors characterize various scheduling penalties arising from multiprocessor task synchronization and propose to colocate

tasks that access common shared resources, thus transforming global resource sharing into local sharing. In Bate and Emberson [2006], task allocation and priority assignment were defined with the purpose of optimizing the extensibility with respect to changes in task computation times. The proposed solution, based on simulated annealing and the maximum amount of change that can be tolerated in the task execution times without missing end-to-end deadlines, was computed by scaling all task times by a constant factor. Also, a model of event-based activation for task and messages was assumed. In [Hamann et al. 2006, 2007], a generalized definition of extensibility on multiple dimensions (including changes in the execution times of tasks, as in our article, but also period speed-ups and possibly other metrics) was presented. Also, a randomized optimization procedure based on a genetic algorithm was proposed to solve the optimization problem. These papers focus on the multi-parameter Pareto optimization and how to discriminate the set of optimal solutions. The main limitation of this approach is complexity and expected running time of the genetic optimization algorithm. In addition, randomized optimization algorithms are difficult to control and give no guarantee on the quality of the obtained solution.

Indeed, in the cited papers, the use of genetic optimization is only demonstrated for small sample cases. In Hamann et al. [2007], the experiments show the optimization of a sample system with 9 tasks and 6 messages. The search space consists of the priority assignments on all processors and on the interconnecting bus. Hence, task allocation (possibly the most complex step) and signal to message packing are not subject to optimization. Yet, a complete robustness optimization takes approximately 900 and 3000 seconds

In Metzner and Herde [2006], a SAT-based approach for task and message placement was proposed. Like our approach, the method provides optimal solutions to the placement and priority assignment. However, it did not consider signal packing. The problem of optimal packing of periodic signals into CAN frames when the transmission of signals is subject to deadline constraints and the optimization metric is the minimization of the bus utilization has been proven to be NP-hard in Sandstrom et al. [2000]. Commercial (the middleware tool by Volcano [Casparsson et al. 1999]) and research solutions [Saket and Navet 2006; Sandstrom et al. 2000] to this problem exist. However, they are all based on the assumption that the designer already allocated the tasks to the ECUs and partitioned the end-to-end deadlines into task and message deadlines.

A direct comparison of our results with those presented in these works is not possible. First, contrary to all of them, we do not start from a task and message model, but from a communication model in which tasks exchange signals. The optimal generation of the message set by signal packing is one of our objectives. Second, the task and message activation model we use is different from the one assumed in some of these works. Others only apply to time-triggered systems. In approaches that include stochastic optimization procedures, such as Racu et al. [2005], Hamann et al. [2006, 2007] and He et al. [2009], details on the mutation operator and the genetic encoding of the system configuration [Racu et al. 2005; Hamann et al. 2006, 2007] and on the transition operator and metric function in [He et al. 2009] are missing so that those procedures cannot be exactly replicated. In this work, we programmed a simulated annealing optimizer as a term of comparison. Our SA algorithm, while not strictly the same as the one in He et al. [2009], probably exhibits similar performance. Also, it belongs to the same class (stochastic optimization) of algorithms as the genetic algorithms in the other three papers and is expected to share similar performances and possible disadvantages as discussed in the following paragraphs.

Our *mixed integer linear programming* (MILP) formulation for this problem, as opposed to heuristic search methods or stochastic optimization methods has the following advantages.

—*It provides a measure of the solution quality with respect to the optimum cost*. The distance of the current solution from the optimum can be bounded at any time (computed as the gap between the solutions of the primal and dual problem) and allows us to evaluate the quality of the intermediate solutions generated by the solver. The method also provides a possible guarantee of optimality in case the solver terminates and finds the optimum solution. This guarantee is practically impossible to achieve when using stochastic optimization methods.
—An MILP formulation clearly separates the optimization function from the feasibility constraints. This makes retargeting to a different optimization metric easier. Also, most solvers can handle the more general class of convex functions so the application is not strictly limited to linear metrics (but dealing with non-convex functions can be an issue).
—It can also easily accommodate additional constraints or legacy components that make some design variables constants in the formulation.
—For computing the solution, it is possible to leverage the availability of solvers that have been designed and programmed for good runtime and space performance.
—The solution is more easily formulated for reuse by other designers who want to adopt it.

Finally, this article is an extended and revised version of our previous work [Zheng et al. 2007]. With respect to it, we extended the formulation to include the possibility of gateways and therefore extended the range of applicability to multi-bus systems. We also extended the applicability of the method to more complex systems (including those with multiple buses) by developing several techniques in a multistaged approach in which the results of the first optimization stage are used in the second stage to bound the search space. Consequently, the case study has been extended to include a dual bus system derived from an automotive system with active safety functionality. Also, to compare the results with alternative (stochastic optimization) approaches, we derived a simulated annealing optimization engine for the same problem, and we compared the results and running times of our method with the simulated annealing algorithm. Finally, the proposed formulation shows some similarity with the one we proposed in Zhu et al. [2009]. However, in Zhu et al. [2009] the optimization metric was related to extensibility, and the problem complexity (mostly because of the metric function, which could not be encoded in a linear expression) made it impossible to use MILP except for an early optimization stage, limited to the task placement only. In Zhu et al. [2009], the optimization process heavily relies on heuristics used in a second stage.

2. MODEL AND ASSUMPTIONS

Our system (an example in Figure 2) consists of a *physical architecture*, in which p heterogeneous ECUs $E = \{e_1, e_2, \ldots, e_p\}$ are connected through q controller area network buses $B = \{b_1, b_2, \ldots, b_q\}$, and a *logical architecture* in which n tasks belonging to the set $T = \{\tau_1, \tau_2, \ldots, \tau_n\}$ perform the distributed computations required by the functions. Signals $S = \{s_1, s_2, \ldots, s_m\}$ are exchanged among pairs of tasks. Each signal s_i carries a variable amount of information (encoded in a variable number of bits). β_{s_i} is the length of the signal s_i in bits. The signal exchanged between two tasks is represented as a directed link in the figure, so that the computation flow can be expressed as a directed graph. Multicast signals are represented in our formulation as multiple signals, connecting the source with every possible destination (we set up constraints to avoid having more than one such signal in a bus message. More details will be explained in Section 3.3).

In systems with multiple CAN buses, the source and destination task may not reside on ECUs that connect to the same bus. In this case, a signal exchanged among them goes

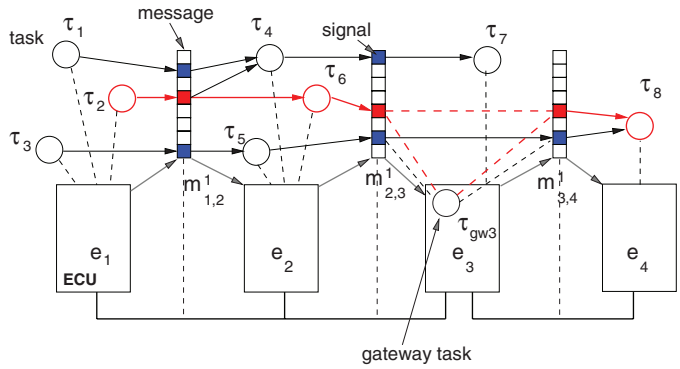

Fig. 2. Mapping of tasks to ECUs and signals to messages.

through a gateway ECU and is forwarded by a gateway task. Gateways are included in our model with some restrictive (but realistic) assumptions.

—Any communication between two ECUs is never going to need more than one gateway hop (every bus is connected to all the others through one gateway ECU). This assumption, realistic for small systems, could probably be removed at the price of additional complexity.
—A single gateway ECU connects any two buses.
—A single task is responsible for signal forwarding on each gateway ECU. This task is fully determined. Note that there might be other regular tasks running on the gateway ECU.

A path p is an ordered interleaving sequence of tasks and signals, defined as $p = [\tau_{r_1}, s_{r_1}, \tau_{r_2}, s_{r_2}, \ldots, s_{r_{k-1}}, \tau_{r_k}]$. $src(p) = \tau_{r_1}$ is the path's source and $snk(p) = \tau_{r_k}$ is its sink. Sources are activated by external events, while sinks activate actuators. Multiple paths may exist between each source-sink pair. We assume all tasks in a path perform computations that contribute to a distributed function, from the collection of sensor data, to remote actuation(s). All computations in a path are also referred to as *end-to-end computations*. The worst-case end-to-end latency incurred when traveling a path p is denoted as l_p. In a computation model in which information is propagated by periodic writing and sampling of values, with tasks in a path possibly having different periods (with possible oversampling and undersampling), the definition of end-to-end latency is subject to ambiguity and deserves better explanation. In our case, we assume a definition related to the practical implications of the end-to-end latency: the largest possible time interval that is required for the change of the input (or sensed) value at one end of the chain to be propagated and cause a value change (or an actuation response) as the output of the last task at the other end of the chain. The path deadline for p, denoted by d_p, is an application requirement that may be imposed on selected paths. We use P to denote the set of all time-sensitive paths with such deadline requirement.

Tasks are executed on the ECUs and activated periodically. The allocation of a task is indicated as a relation A_{τ_i,e_j} meaning that task τ_i is executed on e_j. The period of τ_i is indicated as T_i. At the end of their execution, tasks produce their output signal, which inherits the period of the sender task. We allow the system ECUs to be heterogeneous, but we assume that the worst-case computation time of each task τ_i on each ECU e_j is known or can be estimated at $C_{i,j}$.

After the mapping of the tasks to the ECUs, the signals are mapped into messages exchanged between ECU pairs. $M = \{m^r_{p,q} | e_p, e_q \in E, r = 1 \ldots u^{\max}_{p,q}\}$ is the message set.

$u_{p,q}^{\max}$ is defined as the maximum number of messages between the two ECUs p and q. Given that the allocation of tasks is not known a priori, messages are meant to be as possible containers of signals. When defining optimization variables, their maximum number needs to be estimated or upper bound (the number of signals in the system provides a trivial such bound). All messages are periodic, with period $T_{m_{p,q}^r}$, and are scheduled according to their priority $p_{p,q}^r$ on the CAN bus (as defined by the standard). The mapping rules require that each signal mapped to a message must have the same source and destination ECUs (its transmitter and receiver tasks must be allocated on the source and destination ECU of the message) and the same period of the message.

In CAN, the message size is limited to a maximum of 64 bits. A signal larger than 64 bits should be fragmented and transmitted in multiple messages. In our approach, we don't consider signal fragmentation, but we assume that the length of each signal always allows it to be transmitted in a single message. The designer may perform an a priori fragmentation of larger signals to fit this model.

2.1. Resource Scheduling

The worst-case response time for a periodic task τ_i scheduled with fixed priority in a generic preemptive system can be computed by considering the *busy period* of its priority level starting from the *critical instant* [Liu and Layland 1973] for τ_i. A busy period of a given priority level π_i is a time interval in which the processor is continuously executing tasks of priority level π_i or higher. Also, the critical instant is the release time resulting in the worst-case response time for a given task, which occurs when the task is released at the same time with all other higher priority (periodic) tasks. In this case, the worst-case response time can be computed as the sum of the computation time of the task itself, plus the time interference from higher priority tasks (time spent executing them). If $I_{j,i}(r_i)$ is the interference of higher priority task τ_j on τ_i in a time interval of length r_i, then

$$r_i = C_i + \sum_{j \in hp(i)} I_{j,i}(r_i),$$

where $j \in hp(i)$ spans over all the tasks with higher priority executed on the same ECU as τ_i. Given that

$$I_{j,i} = \left\lceil \frac{r_i}{T_j} \right\rceil C_j,$$

the formula for computing the worst-case response time of the first instance of a generic task τ_i becomes [Joseph and Pandya 1986]

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

when $r_i \leq T_i$ or when the deadline of a task is less than or equal to its period, the previous fixed point formula (1) allows us to compute the worst-case response time of τ_i or, in any case, to check its feasibility against the deadline. Given that the right-hand side of (1) is monotonically increasing, the least value solution r_i (if it exists) can be computed iteratively by starting with $r_i^{(0)} = C_i$, substituting on the right-hand side and computing $r_i^{(1)}$ on the left-hand side, and so on, until a fixed point solution is found or the current value of r_i exceeds the deadline.

When the response time can be larger than the task period, there is no guarantee that the first instance of τ_i activated at the critical instant has the worst-case response time, but all instances of τ_i in the busy period need to be checked. The formula therefore

becomes [Lehoczky 1990]

$$\begin{aligned} &r_i(q) = (q+1)C_i + \sum_{j \in hp(i)} \left\lceil \frac{r_i(q)}{T_j} \right\rceil C_j, \\ &r_i = \max_q \{r_i(q) - qT_i\}, \\ &\text{for all } q = 0 \ldots q^* \text{ until } r_i(q^*) \leq T_i. \end{aligned} \tag{2}$$

The index q refers to all the instances of τ_i inside the busy period. The last instance q^* coincides with the end of the busy period, when $r_i(q^*) \leq T_i$. The maximum among the response times of all of them ($q = 0, \ldots q^*$) is the worst-case response time. Message objects are transmitted over a CAN bus. The evaluation of the worst-case latency for the messages follows the same rules for the worst-case response time of the tasks with the exception that an additional blocking term B_i must be included in the formula in order to account for the nonpreemptability of CAN frames. Of course, in the case of messages, the term C_i indicates the time that is needed for the actual transmission of all the bits of the message frame on the network. If a generic message m_i has a data packet of β_{m_i} bits (obtained by adding up the bit lengths of the signals mapped into it) and a frame overhead of Oh_i (for CAN messages 46 bits), and the bus speed or bit-rate is B_{speed}, then the transmission time C_i is computed as

$$C_i = \frac{\beta_{m_i} + Oh_i}{B_{speed}}.$$

An upper bound of the response time is obtained [Davis et al. 2007] when the blocking term B_i for a generic message m_i is approximated with the largest possible frame transmission time ($w_i > 0$ is the queuing delay part of r_i, without the transmission time). In the general case (response times possibly larger than periods), the formula is

$$\begin{aligned} &w_i(q) = B_i + qC_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i(q)}{T_j} \right\rceil C_j, \\ &r_i = \max_q \{C_i + w_i(q) - qT_i\}, \\ &\text{for all } q = 0 \ldots q^* \text{ until } r_i(q^*) \leq T_i. \end{aligned} \tag{3}$$

A lower bound on w_i and r_i can be computed by only considering the first instance ($q = 0$), as shown in the following text. Once again, the exact response time is obtained using (4) if $r_i \leq T_i$.

$$r_i = B_i + C_i + \sum_{j \in hp(i)} \left\lceil \frac{r_i - C_i}{T_j} \right\rceil C_j. \tag{4}$$

2.2. Periodic Activation Model

The worst case end-to-end latency can be computed for each path by adding the worst-case response times of all the tasks and global signals on the path, as well as the periods of all the global signals and their destination tasks on the path.

$$l_p = \sum_{\tau_i \in p} r_{\tau_i} + \sum_{s_i \in p \wedge s_i \in GS} \left(r_{s_i} + T_{s_i} + T_{dst_{s_i}} \right), \tag{5}$$

where GS is the set of all global signals. Of course, in the case where gateways are used across buses, the signals to and from possible gateway tasks, as well as the response time of the gateway task itself and the associated sampling delays must be included in the analysis.

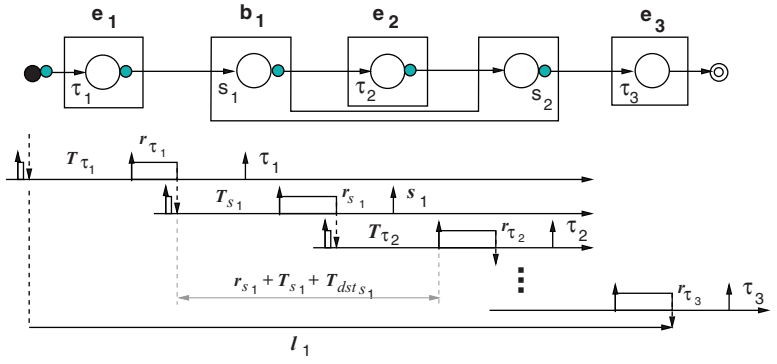

Fig. 3. The periodic activation model and the end-to-end latency on a path.

We need to include periods of global signals and their destination tasks because of the asynchronous sampling of communication data. Figure 3 shows a sequence of task and message activations and response times that results in the longest end-to-end latency. The upper part of the figure represents a computation and communication path of three tasks (τ_1, τ_2, τ_3) on three ECUs (e_1, e_2, and e_3, respectively) and two signals, s_1 from τ_1 to τ_2 and s_2 from τ_2 to τ_3. In the worst case, as shown in Figure 3, we assume an event occurs on the environment, and it is sensed by τ_1. The event occurs immediately after the completion of an instance of τ_1 with almost nil response time. Hence, it will be read by the next instance of the task, and the result will be produced after its worst-case response time, that is, after $T_{\tau_1} + r_{\tau_1}$ time units. τ_1 produces information derived from the event processing that is encoded in the signal s_1. However, the message transmitting s_1 is sent by a message that we assume is enqueued right before the signal is produced. Hence, one message period must be added to the latency, together with its response time. The same reasoning applies to the execution of all tasks that are the destinations of global signals and applies to global signals themselves. However, for local signals, the destination task can be activated with a phase equal to the worst-case response time of the source task under the condition that their periods are harmonic, which is almost always true in practical designs. In this case, we only need to add the response time of the destination task. Similarly, it is sometimes possible to synchronize the queuing of a message for transmission with the execution of the source tasks of the signals present in that message. This would reduce the worst-case sampling period for the message transmission and decrease the latency in Equation (5). In this work, we do not consider these possible optimizations and leave them to future extensions.

3. OBJECTIVE AND FORMULATION

The objective of our design problem is to find the best possible

—allocation of tasks onto the ECUs,
—packing of signals to messages on multiple buses,
—assignment of priorities to tasks and messages,

given

—constraints on (some) end-to-end latencies,
—constraints on the message size,

with respect to the

—minimization of a set of end-to-end latencies.

We formulate our problem in the general framework of mathematical programming (MP), where the system is represented with parameters, decision variables, and constraints over the parameters and decision variables. An objective function, defined over the same set of variables, characterizes the optimal solution. Our problem allows a MILP formulation that is amenable to automatic processing. According to the definition [Boyd et al. 2006], an MILP program in standard form is:

$$\text{minimize} \quad c^T x \tag{6}$$

$$\text{subject to} \quad Ax = b \tag{7}$$

$$x \geq 0, \tag{8}$$

where $x = (x_1, \ldots, x_n)$ is a vector of positive real or integer-valued decision variables. A is an $m \times n$ full-rank constant matrix, with $m < n$, b and c are constant vectors with dimension $n \times 1$. Constraints of the type $Ax \leq b$ can be handled by adding a suitable set of variables, and then transforming such inequalities in the standard form. Once the optimization variables are selected, they define a multidimensional design space. Inside this space, there are values of these variables for which the system is feasible or schedulable against the timing constraints. The region defined by the union of these values is called *feasibility region* (as in Bini and Buttazzo [2002]). MILPs can be solved very efficiently by a variety of solvers. In this work, we make use of the CPLEX solver. CPLEX (now from IBM, formerly from ILOG) is a leading market solution with high performance, several plugins, and optimization options. Further, it is currently available for free for academic and noncommercial use.

The main difficulty of an MILP approach lies in the possible large number of variables and constraints and the resulting large solution time. Further, the feasibility region of tasks and messages with respect to deadlines is not convex and requires a careful representation or approximation. The form of the constraints and objective function must be chosen carefully such that the formulation captures the behavior of the system and yet remains amenable to efficient solving.

3.1. Preface to the Optimization Problem Definition

Task allocation, signal packing, and priority optimization are managed at the same time by the MILP framework. However, while the tasks are mapped into the ECUs and the number and type of ECUs are known at problem definition time, the number, period, and priority of the messages exchanged by the ECUs are unknown in advance. They result from the number and type of the signals that need to be exchanged among ECUs, which depend, in turn, on the task allocation. Our problem formulation requires that we represent messages by a suitable set of variables in order to define the signal to message mapping constraints and the latency of each message (and signal).

Therefore, we bound the number of messages that can be possibly exchanged between any ECU pair and we define a corresponding number of message placeholders acting as possible signal containers. $u_{p,q}^{\max}$ is the upper bound for the number of messages $m_{p,q}^{r}$ between ECU pair e_p and e_q ($r = 1..u_{p,q}^{\max}$). Of course, this means that we need a preprocessing procedure to determine such upper bound for the number of messages between every ECU pair. Furthermore, one set of such messages is needed for each possible period.

Because of the large number of sets, variables, and constraints and, ultimately, for the sake of clarity, in the following sections we explain the optimization constraints, each section referring to a specific aspect of the problem.

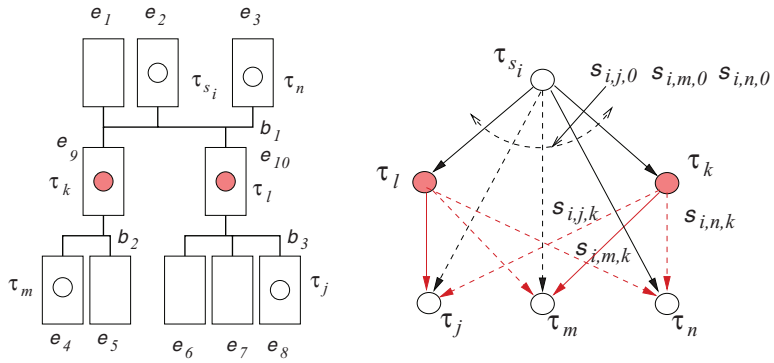

Fig. 4. Signal forwarding by the gateways. A system architecture with a possible task allocation on the left side, and the signal definitions (including gateway signals) on the right side Dashed lines indicate signal variables that are defined 0 (no signal is needed).

3.2. Task to ECU Mapping

3.2.1. Sets and Variables. Based on the problem characterization, the following binary variables are defined

$$A_{\tau_i,e_j} = \begin{cases} 1, & \text{if } \tau_i \text{ is mapped to ECU } e_j \\ 0, & \text{otherwise} \end{cases},$$

$$a_{\tau_i,\tau_j} = \begin{cases} 1, & \text{if } \tau_i \text{ and } \tau_j \text{ are mapped to the same ECU} \\ 0, & \text{otherwise} \end{cases}.$$

3.2.2. Feasibility Constraints. Each task can be mapped to at most one ECU (N constraints).

$$\forall \tau_i, \quad \sum_{e_j \in E} A_{\tau_i,e_j} = 1. \tag{9}$$

Furthermore, there are dependencies among the A_{τ_i,e_j} and the a_{τ_i,τ_j} variables. If tasks τ_i and τ_j are mapped to the same ECU e_k, (10) enforces the variable $a_{\tau_i,\tau_j} = 1$. However, if task τ_i and τ_j are mapped to different ECUs, (11) will set $a_{\tau_i,\tau_j} = 0$.

$$\forall e_k \in E, \quad A_{\tau_i,e_k} + A_{\tau_j,e_k} - 1 \leq a_{\tau_i,\tau_j} \tag{10}$$

$$\forall e_p \neq e_q \in E, \quad 2 - A_{\tau_i,e_p} - A_{\tau_j,e_q} \geq a_{\tau_i,\tau_j} \tag{11}$$

3.3. Signal to Message Mapping

As a result of the task allocation, signals are allocated to one of the message placeholders that are available between any two ECUs.

Gatewaying of signals requires additional definitions and a modification of the signal set to accommodate replicated signals that are sent by the gateway tasks. Gateway tasks are preallocated with known period and priority subject to optimization.

As in Figure 4, for each signal s_i in the task communication model, we use $s_{i,j,0}$ to represent the signal originating from the source task (labeled as τ_{s_i} for simplicity) and directed to the destination τ_j or (if needed) to the gateway task with final destination τ_j. In addition, for each possible gateway τ_k, there is an additional possible signal, labeled $s_{i,j,k}$, representing the signal from the gateway task τ_k to the destination τ_j (allocated on a ECU that can be reached with gateway τ_k).

In case the source task τ_{s_i} and the destination task τ_j are on the same ECU, the signal $s_{i,j,0}$ and all the $s_{i,j,k}$ can be disregarded since they do not contribute to the latency, and

they will not need a message on the bus. In case the source and destination task are connected by a single bus, $s_{i,j,0}$ represents the signal between them, and all the $s_{i,j,k}$ should be disregarded (accomplished by treating them as local signals).

There might be multiple destination tasks for the same source task in the case of multicast. For each s_i, there is one set $s_{i,j,0}$ with as many signals as the number of receivers and one set $s_{i,j,k}$ with cardinality equal to the product of the number of possible gateways by the number of receivers. All gateway signals have the same period and data length of the signals from which they originate.

3.3.1. Sets and Variables. We define following variables:

$$\alpha_{s_{i,j,0},m^r_{p,q}} = \begin{cases} 1, & \text{if } s_{i,j,0} \text{ is mapped to } m^r_{p,q} \\ 0, & \text{otherwise} \end{cases};$$

$$\alpha_{s_{i,j,k},m^r_{p,q}} = \begin{cases} 1, & \text{if } s_{i,j,k} \text{ is mapped to } m^r_{p,q} \\ 0, & \text{otherwise} \end{cases};$$

$$\gamma_{s_{i,j,0},m^r_{p,q}} = \begin{cases} 1, & \text{if } s_{i,j,0} \text{ adds to the length of } m^r_{p,q} \\ 0, & \text{otherwise} \end{cases};$$

$$\gamma_{s_{i,j,k},m^r_{p,q}} = \begin{cases} 1, & \text{if } s_{i,j,k} \text{ adds to the length of } m^r_{p,q} \\ 0, & \text{otherwise} \end{cases}.$$

$\gamma_{s_{i,j,0},m^r_{p,q}}$ and $\gamma_{s_{i,j,k},m^r_{p,q}}$ provide additional information with respect to $\alpha_{s_{i,j,0},m^r_{p,q}}$ and $\alpha_{s_{i,j,k},m^r_{p,q}}$ in the case of multicast signals (e.g., in Figure 4, source task τ_{s_i} has three receivers τ_j, τ_m, and τ_n). In this case, even though our model defines one signal (or two signals when using gateway) for each pair sender-receiver, there is no need to copy the signal multiple times into a message, since CAN messages are broadcast and all remote tasks can read the signal value from the message. $\gamma_{s_{i,j,0},m^r_{p,q}}$ and $\gamma_{s_{i,j,k},m^r_{p,q}}$ are used to nullify multiple copies of such multicast signals.

Finally, since the messages are actually placeholders, and some of them may be empty after the signal mapping stage, we need an additional set of variables

$$Y_{m^r_{p,q}} = \begin{cases} 1, & \text{if message } m^r_{p,q} \text{ is nonempty} \\ 0, & \text{otherwise} \end{cases}$$

to avoid considering those messages in the scheduling and response time computation stage.

3.3.2. Mapping Constraints. Signals must be mapped to messages when communication data must be sent over a bus (sender and receiver tasks are remote).

The boolean variable $a_{s_{i,j,k},b_r}$ is 1 if signal $s_{i,j,k}$ is mapped onto bus b_r and 0 otherwise, similarly for the definition of $a_{s_{i,j,0},b_r}$. To define these set of variables, we need to consider all ECU pairs for each signal from its source to all its destinations τ_j.

$$\forall b_r \in B(e_m), e_m \neq e_p, \quad A_{\tau_{s_i},e_m} + A_{\tau_j,e_p} - 1 \leq a_{s_{i,j,0},b_r} \tag{12}$$

$$\forall B(e_m) \cap B(e_p) = \emptyset, b_r \in B(e_p), \quad A_{\tau_{s_i},e_m} + A_{\tau_j,e_p} - 1 \leq a_{s_{i,j,k},b_r} \tag{13}$$

$$\forall b_r, \quad g_{s_{i,j,0}} \geq a_{s_{i,j,0},b_r} \tag{14}$$

$$g_{s_{i,j,0}} \leq \sum_{b_r} a_{s_{i,j,0},b_r} \tag{15}$$

$$\forall b_r, \quad g_{s_{i,j,k}} \geq a_{s_{i,j,k},b_r} \tag{16}$$

$$g_{s_{i,j,k}} \leq \sum_{b_r} a_{s_{i,j,k},b_r} \tag{17}$$

$$\forall s_{i,j,k}, \quad g_{s_{i,j,k}} \leq g_{s_{i,j,0}}, \tag{18}$$

where $B(e)$ is the set of buses connected to ECU e. The set of constraints defined by (12) for all possible sets (source τ_{s_i}, destination τ_j, source ECU e_m that is on bus b_r, destination ECU e_p that communicates with e_m through b_r) force $a_{s_{i,j,0},b_r}$ to 1 for the bus b_r from e_m to e_p, or from e_m to the gateway task τ_k between e_m and e_p. The following set (13) sets $a_{s_{i,j,k},b_r}$ to 1 (if necessary) for the bus b_r from the gateway to the destination ECU e_p when gatewaying is needed (in this set of constraints, e_p is on b_r, while e_m is not). The variables $a_{s_{i,j,k},b_r}$ have a positive contribution to the cost function, hence they will be set to 0 by the optimization engine, unless forced to 1 by the constraints.

To give an example of these constraints, in Figure 4 the condition for the outgoing signal $s_{i,j,0}$ from τ_{s_i} to τ_j to be on bus b_1, expressed as

$$a_{\tau_{s_i},e_m} + a_{\tau_j,e_p} - 1 \leq a_{s_{i,j,0},b_1},$$

needs to be defined for each pair of ECUs (m, p) as follows: $m \in \{1, 2, 3\}$ and $p \neq m$; $m \in \{9\}$ and $p \in \{1, 2, 3, 10\}$; $m \in \{10\}$ and $p \in \{1, 2, 3, 9\}$. Similar sets of conditions will then need to be defined for b_2 and b_3.

As an example of gatewaying, the condition for the mapping on b_2 of the (possible) signal forwarded by gateway task τ_k on e_9 as part of the communication from τ_{s_i} to τ_j in the figure is expressed by the set.

$$a_{\tau_{s_i},e_m} + a_{\tau_j,e_p} - 1 \leq a_{s_{i,j,k},b_2},$$

defined for all ECU pairs (m, p) where $m = \{1, 2, 3, 10\}$ and $p = \{4, 5\}$.

The value of the boolean variable $g_{s_{i,j,0}}$ is 1 if $s_{i,j,0}$ is a global signal (i.e., transferred on bus), and 0 otherwise. Similarly, $g_{s_{i,j,k}}$ is 1 if the signal $s_{i,j,k}$ (s_i forwarded by gateway τ_k) is global. The definition of $g_{s_{i,j,0}}$ and $g_{s_{i,j,k}}$ is provided by constraints (14)–(15) and (16)–(17), respectively. Finally, $g_{s_{i,j,0}}$ must be 1 if at least one $g_{s_{i,j,k}}$ is 1, as in constraint (18).

Having defined the preceding variables to represent the globality of signals, we have the following constraints to enforce that each signal is mapped to at most one message (or not mapped to any, when communication is local), as in (19) and (20). For all messages $m^r_{p,q}$ (the range of r is constrained by the pair of ECU indexes p, q) and all signals $s_{i,j,0}$,

$$\sum_{r \leq u^{\max}_{p,q}, p,q \in E} \alpha_{s_{i,j,0},m^r_{p,q}} \leq g_{s_{i,j,0}} \tag{19}$$

$$\sum_{r \leq u^{\max}_{p,q}, p,q \in E} \alpha_{s_{i,j,k},m^r_{p,q}} \leq g_{s_{i,j,k}}. \tag{20}$$

Signal $s_{i,j,0}$ is exchanged between task τ_{s_i} and τ_j. If tasks τ_{s_i} and τ_j are mapped, respectively, to ECUs e_p and e_q ($p \neq q$) on the same bus, the signal must be mapped to one of the messages between e_p and e_q, as in (21).

$$\forall e_p \neq e_q, \quad A_{\tau_{s_i},e_p} + A_{\tau_j,e_q} - 1 \leq \sum_{r \leq u^{\max}_{p,q}} \alpha_{s_{i,j,0},m^r_{p,q}}. \tag{21}$$

If task τ_{s_i} and τ_j are mapped to ECUs e_p and e_q that are on different buses and connected through gateway ECU e_g, we have the following constraints (22) and (23) for $s_{i,j,0}$ and

$s_{i,j,k}$.

$$\forall B(e_m) \cap B(e_p) = \emptyset, e_g \text{ gateway}, \quad A_{\tau_{s_i},e_p} + A_{\tau_j,e_q} - 1 \leq \sum_{r \leq u^{\max}_{p,g}} \alpha_{s_{i,j,0},m^r_{p,g}} \tag{22}$$

$$\forall B(e_m) \cap B(e_p) = \emptyset, e_g \text{ gateway}, \quad A_{\tau_{s_i},e_p} + A_{\tau_j,e_q} - 1 \leq \sum_{r \leq u^{\max}_{g,q}} \alpha_{s_{i,j,k},m^r_{g,q}} \tag{23}$$

For multicast signals, different receivers can share the same signal because of the broadcast nature of CAN bus. This is reflected in constraints (24) to (26). The message length in bit $\beta_{m^r_{p,q}}$ is computed by adding up the bits of all the signals mapped into it, as shown in (27), where the data content of each message is constrained to be less than 64 bits, the maximum allowed size (by the protocol) for a CAN message. Note that $\beta_{s_{i,j,0}} = \beta_{s_{i,j,k}} = \beta_{s_i}$.

$$\sum_j \gamma_{s_{i,j,0},m^r_{p,q}} + \sum_{j,k} \gamma_{s_{i,j,k},m^r_{p,q}} \leq 1 \tag{24}$$

$$\forall j \quad \sum_j \gamma_{s_{i,j,0},m^r_{p,q}} \geq \alpha_{s_{i,j,0},m^r_{p,q}} \tag{25}$$

$$\forall j \quad \sum_{j,k} \gamma_{s_{i,j,k},m^r_{p,q}} \geq \alpha_{s_{i,j,k},m^r_{p,q}} \tag{26}$$

$$\sum_{s_{i,j,0} \in S} \gamma_{s_{i,j,0},m^r_{p,q}} \beta_{s_{i,j,0}} + \sum_{s_{i,j,k} \in S} \gamma_{s_{i,j,k},m^r_{p,q}} \beta_{s_{i,j,k}} = \beta_{m^r_{p,q}} \leq 64 \tag{27}$$

Finally, we need to constrain the $Y_{m^r_{p,q}}$ variables that define whether a message has at least one signal or is empty.

$$\forall s_{i,j,0}, \quad Y_{m^r_{p,q}} \geq \alpha_{s_{i,j,0},m^r_{p,q}} \tag{28}$$

$$\forall s_{i,j,k}, \quad Y_{m^r_{p,q}} \geq \alpha_{s_{i,j,k},m^r_{p,q}} \tag{29}$$

$$Y_{m^r_{p,q}} \leq \sum_{s_{i,j,0} \in S} \alpha_{s_{i,j,0},m^r_{p,q}} + \sum_{s_{i,j,k} \in S} \alpha_{s_{i,j,k},m^r_{p,q}} \tag{30}$$

3.4. Worst-Case Response Time of Tasks

For each pair of tasks (τ_i, τ_j), we define

$$p_{i,j} = \begin{cases} 1, & \text{if task } \tau_i \text{ has higher priority than } \tau_j \\ 0, & \text{otherwise} \end{cases}.$$

For the antisymmetric and transitive properties of the priority order relation, it must be (we assume no two task have the same priority level.)

$$\forall i \neq j, \quad p_{i,j} + p_{j,i} = 1 \tag{31}$$

$$\forall i \neq j \neq k, \quad p_{i,j} + p_{j,k} - 1 \leq p_{i,k} \tag{32}$$

The formula that allows us to compute the worst-case response time of a task τ_i is

$$r_i = C_i + \sum_{j \in hp(i)} I_{j,i} C_j,$$

where $hp(i)$ spans over the set of all the higher priority tasks that are allocated on the same ECU as τ_i, and $I_{j,i}$ is the number of interferences of τ_j on τ_i during its response

time.

$$I_{j,i} = \left\lceil \frac{r_i}{T_j} \right\rceil$$

To compute r_i in our MILP framework, we start by adding the following variable

$$y_{i,k} = \begin{cases} n \in \mathbb{N}^+, & \text{number of possible interferences of } \tau_k \text{ on } \tau_i \\ 0, & \text{otherwise.} \end{cases}$$

The definition of the possible number of interferences as function of the response times and periods is captured by

$$0 \leq y_{i,k} - \frac{r_i}{T_k} < 1. \tag{33}$$

In addition, we define

$$x_{i,k} = \begin{cases} n \in \mathbb{N}^+, & \text{number of possible interferences of } \tau_k \text{ on } \tau_i \text{ if } p_{k,i} = 1 \\ 0, & \text{otherwise.} \end{cases}$$

The encoding of this set (as well as others) of conditional constraints in an MILP formulation is performed using the well-known "big-M" formulation. Formally, a set of disjunctive constraints

$$\bigvee_{v \in \mathcal{V}} g(x, v) \leq 0$$

is equivalent to a set of (conjunctive) constraints

$$g(x, v) \leq M_v(1 - \beta_v) \text{ with } \quad v \in \mathcal{V}, \quad \sum_{v \in \mathcal{V}} \beta_v = 1,$$

where β_v is a binary variable and M_v is a vector of upper bounds on the values of the $g(x, v)$ functions. $x_{i,k}$ can be defined in terms of $y_{i,k}$ and $p_{k,i}$ as follows.

$$y_{i,k} - (1 - p_{k,i}) \times M \leq x_{i,k} \leq y_{i,k} \tag{34}$$

$$0 \leq x_{i,k} \leq p_{k,i} \times M \tag{35}$$

Furthermore, to take into account the allocation condition, we need to define also

$$w_{i,k} = \begin{cases} n \in \mathbb{N}^+, & \text{number of possible interferences of } \tau_k \text{ on } \tau_i \text{ if } p_{k,i} = 1 \\ & \text{when they are on the same ECU}(a_{\tau_i,\tau_k} = 1) \\ 0, & \text{otherwise} \end{cases}$$

and

$$z_{i,j,k} = \begin{cases} n \in \mathbb{N}^+, & \text{number of possible interferences of } \tau_k \text{ on } \tau_i \text{ if } p_{k,i} = 1 \\ & \text{when they are on ECU } e_j \\ 0, & \text{otherwise} \end{cases}.$$

Please note that $w_{i,k} \neq 0$ is the only case in which τ_k can actually preempt (i.e., interfere with) τ_i. An additional variable $z_{i,j,k}$ is used to put this information in the context of a given ECU (e_j). These variables can be computed from the previous ones as

$$x_{i,k} - (1 - a_{\tau_i,\tau_k}) \times M \leq w_{i,k} \leq x_{i,k} \tag{36}$$

$$0 \leq w_{i,k} \leq a_{\tau_i,\tau_k} \times M \tag{37}$$

for $w_{i,k}$, and

$$w_{i,k} - (1 - A_{\tau_k,e_j}) \times M \leq z_{i,j,k} \leq w_{i,k} \tag{38}$$

$$0 \leq z_{i,j,k} \leq A_{\tau_k,e_j} \times M \tag{39}$$

for $z_{i,j,k}$.

Finally, the response time of task τ_i (an additional variable $r_i \in \mathbb{R}^+$) can be computed as

$$r_i = \sum_j A_{\tau_i,e_j} C_{i,j} + \sum_k \sum_j z_{i,j,k} C_{k,j}. \tag{40}$$

The formulation of the response time using integer variables has been proposed in several research works, including Zheng et al. [2007] and Metzner and Herde [2006] where the feasibility problem is part of a design optimization based on a SAT-solver. The corresponding feasibility test is exact, but requires a possibly large number of integer variables (in the order of n^2, where n is the number of tasks). Solving the MILP problem in feasible time may be very difficult for medium-sized and sometimes small-sized systems. A simple linear upper bound $r_i^{\uparrow}$ can be obtained by upper bounding the ceiling function and using a real value for $I_{j,i}^{\uparrow}$.

$$I_{j,i}^{\uparrow} = \frac{r_i}{T_j} + 1.$$

A tighter upper bound can be obtained by using the definition of the *load executed at higher priority* provided in Bini et al. [2009]. Because of space limitations, we only present the response-time upper bound without providing further details.

$$r_i^{\uparrow} - \sum_{j \in hp(i)} \left(U_j r_i^{\uparrow} + C_j (1 - U_j) \right) = C_i \tag{41}$$

3.5. Worst-Case Response Time of Messages

For each pair of messages $(m_{p,q}^r, m_{s,t}^f)$, we define

$$p_{m_{p,q}^r, m_{s,t}^f} = \begin{cases} 1, & \text{if } m_{p,q}^r \text{ has higher priority than } m_{s,t}^f \\ 0, & \text{otherwise} \end{cases}.$$

For the antisymmetric and transitive properties of the priority order relation, it must be (we assume no two messages have the same priority level.)

$$p_{m_{p,q}^r, m_{s,t}^f} + p_{m_{s,t}^f, m_{p,q}^r} = 1 \tag{42}$$

$$p_{m_{p,q}^r, m_{s,t}^f} + p_{m_{s,t}^f, m_{x,y}^z} - 1 \leq p_{m_{p,q}^r, m_{x,y}^z}. \tag{43}$$

As a result of the optimization, a priority level is assigned to all messages, including empty placeholders, and it is possible that a high priority level is assigned to one of the empty messages. An additional constraint ensures that this never happens. In (44) message $m_{p,q}^r$, when nonempty ($Y_{m_{p,q}^r} = 1$), has higher priority than the empty message $m_{s,t}^f$ ($Y_{m_{s,t}^f} = 0$). This ensures that empty messages don't contribute to the interference of nonempty messages.

$$Y_{m_{p,q}^r} - Y_{m_{s,t}^f} \times M \leq p_{m_{p,q}^f, m_{s,t}^r} \tag{44}$$

The formula that allows us to upper bound the worst-case response time of a message $m_{p,q}^r$ is

$$r_{m_{p,q}^r} = C_{m_{p,q}^r} + q_{m_{p,q}^r}$$

$$q_{m^r_{p,q}} = B_i + \sum_{j \in hp(i)} I_{m^f_{s,t}, m^r_{p,q}} C_j,$$

where $hp(i)$ spans over the indexes of all the messages with priority higher than $r_{m^r_{p,q}}$, and $I_{m^f_{s,t}, m^r_{p,q}}$ is the number of interferences of $m^f_{s,t}$ on $m^r_{p,q}$ during its queuing time. It is

$$I_{m^f_{s,t}, m^r_{p,q}} = \left\lceil \frac{q_{m^r_{p,q}}}{T^r_{p,q}} \right\rceil.$$

Note that we only consider messages $m^f_{s,t}$ that are on the same bus with $m^r_{p,q}$ (this information can be deduced from which buses ECU p, q, s, and t are connected to).

Furthermore, the transmission times of messages depend upon their lengths according to

$$C_{m^r_{p,q}} = \frac{\beta_{m^r_{p,q}} + 46}{B_{speed}} = \frac{\beta_{m^r_{p,q}}}{B_{speed}} + \frac{46}{B_{speed}},$$

where $\beta_{m^r_{p,q}}$ is the size of the message (total number of bits mapped into it), and 46 is the number of protocol bits in a CAN frame.

Similar to the computation of the response times of the tasks, the additional variables $z_{m^r_{p,q}, m^f_{s,t}, s_{i,j,0}}$, $z_{m^r_{p,q}, m^f_{s,t}, s_{i,j,k}}$, $w_{m^r_{p,q}, m^f_{s,t}}$, $x_{m^r_{p,q}, m^f_{s,t}}$, and $y_{m^r_{p,q}, m^f_{s,t}}$ are defined.

The possible number of interferences $y_{m^r_{p,q}, m^f_{s,t}}$ is obtained from

$$0 \leq y_{m^r_{p,q}, m^f_{s,t}} - \left(r_{m^r_{p,q}} - \frac{\beta_{m^r_{p,q}} + 46}{B_{speed}} \right) \Big/ T_{m^f_{s,t}} < 1. \tag{45}$$

$x_{m^r_{p,q}, m^f_{s,t}}$, the possible number of interferences of a higher priority message $m^r_{s,t}$ on $m^f_{p,q}$ can be computed from $y_{m^r_{p,q}, m^f_{s,t}}$.

$$y_{m^r_{p,q}, m^f_{s,t}} - \left(1 - p_{m^r_{s,t}, m^f_{p,q}}\right) \times M \leq x_{m^r_{p,q}, m^f_{s,t}} \tag{46}$$

$$x_{m^r_{p,q}, m^f_{s,t}} \leq y_{m^r_{p,q}, m^f_{s,t}} \tag{47}$$

$$0 \leq x_{m^r_{p,q}, m^f_{s,t}} \leq p_{m^r_{s,t}, m^f_{p,q}} \times M \tag{48}$$

$w_{m^r_{p,q}, m^f_{s,t}}$ is computed from $x_{m^r_{p,q}, m^f_{s,t}}$ and represents the number of interferences from a higher priority nonempty message $m^r_{s,t}$ (the only ones of practical relevance).

$$x_{m^r_{p,q}, m^f_{s,t}} - \left(1 - Y_{m^f_{s,t}}\right) \times M \leq w_{m^r_{p,q}, m^f_{s,t}} \tag{49}$$

$$w_{m^r_{p,q}, m^f_{s,t}} \leq x_{m^r_{p,q}, m^f_{s,t}} \tag{50}$$

$$0 \leq w_{m^r_{p,q}, m^f_{s,t}} \leq Y_{m^f_{s,t}} \times M \tag{51}$$

and $z_{m^r_{p,q}, m^f_{s,t}, s_{i,j,0}}$, $z_{m^r_{p,q}, m^f_{s,t}, s_{i,j,k}}$ from $w_{m^r_{p,q}, m^f_{s,t}}$, $\gamma_{s_{i,j,0}, m^r_{s,t}}$ and $\gamma_{s_{i,j,k}, m^r_{s,t}}$

$$w_{m^r_{p,q}, m^f_{s,t}} - \left(1 - \gamma_{s_{i,j,0}, m^f_{s,t}}\right) \times M \leq z_{m^r_{p,q}, m^f_{s,t}, s_{i,j,0}} \tag{52}$$

$$z_{m^r_{p,q}, m^f_{s,t}, s_{i,j,0}} \leq w_{m^r_{p,q}, m^f_{s,t}} \tag{53}$$

$$0 \leq z_{m^r_{p,q}, m^f_{s,t}, s_{i,j,0}} \leq \gamma_{s_{i,j,0}, m^r_{s,t}} \times M \tag{54}$$

$$w_{m^r_{p,q}, m^f_{s,t}} - \left(1 - \gamma_{s_{i,j,k}, m^r_{s,t}}\right) \times M \leq z_{m^r_{p,q}, m^f_{s,t}, s_{i,j,k}} \tag{55}$$

$$z_{m^r_{p,q}, m^f_{s,t}, s_{i,j,k}} \leq w_{m^r_{p,q}, m^f_{s,t}} \tag{56}$$

$$0 \leq z_{m^r_{p,q},m^f_{s,t},s_{i,j,k}} \leq \gamma_{s_{i,j,k},m^r_{s,t}} \times M \tag{57}$$

Finally, (58) computes the bound for the worst-case response time $r_{m^r_{p,q}} \in \mathbb{R}^+$ of message $m^r_{p,q}$ based on the number of interference from other messages

$$\begin{aligned} r_{m^r_{p,q}} \;\leq\; & \frac{\beta_{m^r_{p,q}} + 46}{B_{speed}} + B_{m^r_{p,q}} + \frac{1}{B_{speed}} \sum_{m^f_{s,t}} \sum_{s_{i,j,0}} z_{m^r_{p,q},m^f_{s,t},s_{i,j,0}} \beta_{s_{i,j,0}} \\ & + \frac{1}{B_{speed}} \sum_{m^f_{s,t}} \sum_{s_{i,j,k}} z_{m^r_{p,q},m^f_{s,t},s_{i,j,k}} \beta_{s_{i,j,k}} + \frac{46}{B_{speed}} \sum_{m^f_{s,t}} w_{m^r_{p,q},m^f_{s,t}}, \end{aligned} \tag{58}$$

where each $m_{s,t}$ is on the same bus with $m_{p,q}$. And we ensure that $r_{m^r_{p,q}} > 0$ only when $Y_{m^r_{p,q}} \neq 0$.

$$\begin{aligned} & \frac{\beta_{m^r_{p,q}} + 46}{B_{speed}} + B_{m^r_{p,q}} + \frac{1}{B_{speed}} \sum_{m^f_{s,t}} \sum_{s_{i,j,0}} z_{m^r_{p,q},m^f_{s,t},s_{i,j,0}} \beta_{s_{i,j,0}} \\ & + \frac{1}{B_{speed}} \sum_{m^f_{s,t}} \sum_{s_{i,j,k}} z_{m^r_{p,q},m^f_{s,t},s_{i,j,k}} \beta_{s_{i,j,k}} \\ & + \frac{46}{B_{speed}} \sum_{m^f_{s,t}} w_{m^r_{p,q},m^f_{s,t}} + \left(Y_{m^r_{p,q}} - 1\right) \times M \leq r_{m^r_{p,q}} \end{aligned} \tag{59}$$

$$r_{m^r_{p,q}} \leq Y_{m^r_{p,q}} \times M \tag{60}$$

Paths are defined on the tasks and the communication signals between them. the following constraints bound the latency of a message to the latency of all the signals that are mapped into it. The response time of signal $s_{i,j,0}$ and $s_{i,j,k}$ are denoted as $r_{s_{i,j,0}}$ and $r_{s_{i,j,0}}$, respectively.

$$r_{m^r_{p,q}} - \left(1 - \alpha_{s_{i,j,0},m^r_{p,q}}\right) \times M \leq r_{s_{i,j,0}} \tag{61}$$

$$r_{s_{i,j,0}} \leq r_{m^r_{p,q}} + (1 - \alpha_{s_{i,j,0},m^r_{p,q}}) \times M \tag{62}$$

$$r_{s_{i,j,0}} \leq g_{s_{i,j,0}} \times M \tag{63}$$

$$r_{m^r_{p,q}} - \left(1 - \alpha_{s_{i,j,k},m^r_{p,q}}\right) \times M \leq r_{s_{i,j,k}} \tag{64}$$

$$r_{s_{i,j,k}} \leq r_{m^r_{p,q}} + \left(1 - \alpha_{s_{i,j,k},m^r_{p,q}}\right) \times M \tag{65}$$

$$r_{s_{i,j,k}} \leq g_{s_{i,j,k}} \times M \tag{66}$$

$$\tag{67}$$

If τ_{s_i} and τ_j are mapped to different ECUs, (61) and (62) bounds the signal latency $r_{s_{i,j,0}}$ to the latency of message $r_{m^r_{p,q}}$. However, inequality (63) forces $r_{s_{i,j,0}}$ to 0 when the signal is local, that is, when τ_{s_i} and τ_j are mapped to the same ECU. Similar constraints are defined for $r_{s_{i,j,k}}$.

3.6. Worst-Case End-to-End Latency

We are now ready to compute the end-to-end latency. For path $p \in P$, its latency l_p is defined as

$$\forall p \in P, \quad l_p \;=\; \sum_{\tau_i \in p} r_{\tau_i} + \sum_{s_i \in p} (r_{s_{i,j,0}} + g_{s_{i,j,0}} T_{s_i} + g_{s_{i,j,0}} T_{\tau_j}$$

$$+\sum_{k}(r_{s_{i,j,k}}+g_{s_{i,j,k}}T_{s_i}+g_{s_{i,j,k}}T_{\tau_k}+g_{s_{i,j,k}}C_{\tau_k})) \tag{68}$$

$$\forall p \in P, \quad l_p \leq d_p, \tag{69}$$

where τ_i is the generic task and s_i is the generic signal on the path p, and we assume the response time for gateway task τ_k is C_{τ_k} (i.e., a gateway task has the highest priority on its ECU). Latencies on the paths should be no greater than the deadline d_p (69).

3.7. Objective Function

The design problem we described in this article was originally formulated as a feasibility problem. However, in the definition of the software architecture of the system there are clearly performance objectives that can be optimized. The target platforms that motivated this work are automotive architectures supporting *active safety* functions. These functions make use of (physically distributed) sensors of different types to understand and evaluate the environment surrounding the car, detect possible sources of danger, and inform the driver or actuate the car systems (brakes, steer, throttle). The actuations are performed as an overlay to the driver's actions with the objective of preventing collisions or other types of accidents. In these systems, computation paths refer to the set of computations occurring from the sensor detection of environment events and objects to the actuation action (such as braking or sounding a chime that alerts the driver). Clearly, the performance of these functions improves when the reaction (or response) times are shorter. Given the set of the latencies over all time-sensitive paths, one possible definition of the objective function is to minimize the sum of these end-to-end latencies as an attempt at generically shortening the response times over all the time-sensitive paths.

$$\min \sum_{p \in P} l_p \tag{70}$$

This metric function is of course quite coarse. It does not target individual paths and it possibly allows some responses to be quite large (but still within the deadlines imposed as constraints), while others can be very small.

An alternative objective function can be defined with the purpose of maximizing the distance between the response times and the deadlines on a set of given paths. This metric not only tends to shorten response times on those paths, but also more directly relates to the objectives of improving robustness and extensibility. By increasing the slack time between response times and deadlines, we achieve better protection against errors in the estimates of the C_i and allow for better extensibility when more tasks need to be added to the system or some of the (control) functions become more complex and require some additional computation time. Formally, the objective in this case is minimizing the maximum lateness (defined as the path latency minus its deadline) among the set of time-sensitive paths.

$$\min \max_{p \in P} \{l_p - d_p\} \tag{71}$$

In the following experimental section, we are going to discuss the results that we obtained by applying this function (together with the previous one) on a case study of industrial complexity and its extension to larger cases to estimate the scalability of the approach.

3.8. Notation Types

We summarize the types of the notations used in the formulation in Table I.

Table I. Notation Types in MILP Formulation

Type	Notations
Boolean variables	A_{τ_i,e_j}, α_{τ_i,τ_j}, $\alpha_{s_{i,j,0},m^r_{p,q}}$, $\alpha_{s_{i,j,k},m^r_{p,q}}$, $\gamma_{s_{i,j,0},m^r_{p,q}}$, $\gamma_{s_{i,j,k},m^r_{p,q}}$, $Y_{m^r_{p,q}}$, $a_{s_{i,j,0},b_r}$, $a_{s_{i,j,k},b_r}$, $g_{s_{i,j,0}}$, $g_{s_{i,j,k}}$, $p_{i,j}$, $p_{m^r_{p,q},m^f_{s,t}}$,
Integer variables	$\beta_{m^r_{p,q}}$, $y_{i,k}$, $x_{i,k}$, $w_{i,k}$, $z_{i,j,k}$, $z_{m^r_{p,q},m^f_{s,t},s_{i,j,0}}$, $z_{m^r_{p,q},m^f_{s,t},s_{i,j,k}}$, $w_{m^r_{p,q},m^f_{s,t}}$, $x_{m^r_{p,q},m^f_{s,t}}$, $y_{m^r_{p,q},m^f_{s,t}}$,
Real variables	r_i, l_p, $C_{m^r_{p,q}}$, $r_{s_{i,j,0}}$, $r_{s_{i,j,k}}$, $r_{m^r_{p,q}}$
Integer Constants	$u^{\max}_{p,q}$, $\beta_{s_{i,j,0}}$, $\beta_{s_{i,j,k}}$, β_{s_i},
Real Constants	$C_{i,j}$, T_i, d_p, M, B_{speed}, $T_{m^f_{s,t}}$, $B_{m^r_{p,q}}$

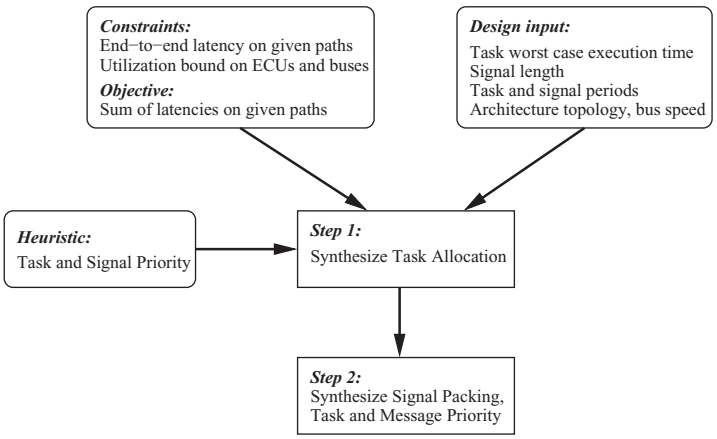

Fig. 5. Two-step synthesis approach.

4. SYNTHESIS STEPS

Section 3 describes an integrated formulation for task allocation, signal packing, as well as task and message priority optimization. This problem formulation provides an optimal solution when solvable. However, the complexity is typically too high for the sizes of industrial applications because of the large number of integer (for task and message feasibility) and binary (for priority assignment and allocation of task and messages) variables.

Several strategies can be used to cope with this complexity, giving up the guarantee of finding the optimal solution for acceptable running times and possibly good quality solutions. As a general framework, we propose a faster approximated two-step synthesis method, as shown in Figure 5. The problem is divided into two subproblems. At each step, the subproblem is formulated as an MILP based on the variables and constraints defined in Section 3, then solved by mathematical programming tools.

In Step 1, we assume that one message is reserved for each signal, and the priorities of tasks and one-signal messages are assigned according to their periods by a preprocessing heuristic based on the Rate Monotonic policy. In the first subproblem, we synthesize the task allocation to optimize the sum of the latencies of given paths, while also satisfying the deadline constraints on those paths.

In Step 2, we use the task allocation result from Step 1 and synthesize signal packing, message priority, and task priority. The objective is still to optimize the sum of the latencies of given paths, while satisfying the deadline constraints on paths and the constraints on message size.

However, when the system has more than one CAN bus and a sufficiently large number of ECUs and tasks (like one of our case studies shown in next section), the complexity is excessive even for Step 1 and a two-stage optimization is used for this step.

Also in one of our case studies, we attempted optimizing both task allocation and priority in Step 1 and reoptimizing the task priority with signal packing and message priority in Step 2, with results shown in next section.

5. CASE STUDY: AN EXPERIMENTAL VEHICLE SUBSYSTEM

We demonstrated the applicability and the possible benefits of our approach with a case study derived from a subsystem of an experimental vehicle that incorporates advanced active safety functions. The vehicle supports distributed functions with end-to-end computations collecting data from 360° sensors to the actuators, consisting of the throttle, brake, and steering subsystems and of advanced HMI (Human-Machine Interface) devices. Examples of active safety functions, are Adaptive Cruise Control (ACC), Lane Departure Warning or Lane Keeping Systems. In an active cruise control system, a set of radars (or other sensors) scans the road in front of the car to ensure that there are no other cars or objects moving at a lower speed or even stopped in the middle of the lane. If such an object is detected, the system lowers the target speed of the cruise control until it matches the speed of the detected obstacle.

These functions are deployed together in a car electronics system, sharing the sensing and actuation layers and possibly also intermediate processing stages, such as the sensor fusion and object detection functions or the actuator arbitration layers. The result is a complex graph of functions (programmed as tasks) with a high degree of communication dependency and deadlines on selected pairs of endpoints.

In our case study, the system is a subset of those functions, with their tasks and communication signals. Also, we focused our analysis on a subset of the architecture ECUs (those considered for the functions of interest). For our study, we removed allocation constraints that actually apply to the real system (e.g., sensor processing stages are bound onto the same ECU to which the sensor is connected) that make the problem easier in reality.

We applied our method to two different platform options. In both cases, the subsystem that is the subject of our study consists of a total of 41 tasks and 83 CAN signals exchanged among them. Worst-case execution time estimates have been obtained for all tasks, and the bit length of the signals is between 1 (for binary information) and 64 (full CAN message).

End-to-end deadlines are imposed over 10 pairs of source-sink tasks in the system with 171 possible computation paths having them as endpoints. The deadline is set at 300ms for 8 source-sink pairs and 100ms for the other two. For challenging the applicability of the method to our case study, we included in the formulation of the optimization problem all tasks and signals, not only those involved in end-to-end computations with deadlines. In reality, the problem could have been made easier by assigning the lowest priority levels to tasks not involved in time-sensitive functions, and complexity could have been reduced by only optimizing the tasks and signals that are part of paths with deadlines. The remaining tasks and signals can be assigned lower priorities and allocated to ECUs and messages based on other considerations (possibly load balancing) in such a way that they do not interfere with the latencies of the critical paths. Even in large systems, the actual amount of processing and communication that is subject to hard deadlines or for which performance depends on time can be only a small fraction.

The architecture platform of our target system consists of 9 ECUs. For the purpose of our experiments, we assumed all ECUs have the same computation power, which is not actually true in the real system. This simplification does not affect the complexity

Table II. Result of the Single-Bus Configuration

Step	#Var	#Int	#Binary	#Con	Time(s)	Result(ms)	Gap
Step 1	34075	0	450	80817	18539.9	12295.3	Opt
Step 2	3247	1122	1841	18946	0.45	4927.5	Opt

or the running time of the optimization algorithm in any way and is only motivated by the lack of WCET (worst-case execution time) data for the tasks on all possible ECUs. Finally, for extensibility reasons, the utilization upper bound of each ECU and bus has been set to 70%.

For both platform options, the experiments are performed on a 3.2GHz processor with 2GB RAM, using CPLEX 11.0 as the MILP solver.

In the following Sections 5.1 and 5.2, the objective function for optimization is the sum of end-to-end latencies as defined in (70). In Section 5.3, the alternative objective of maximum lateness is explored as defined in (71).

5.1. First Option: Single Bus Configuration

In the first configuration option, the ECUs are connected with a single CAN bus at 500kb/s.

In Step 1, to reduce the complexity, we optimized the task allocation using the upper bound formulation (41) from Bini et al. [2009] to approximate the response times of tasks and messages. This significantly reduces the number of integer variables, while ensuring the results we get are correct system configurations. We also relaxed some integer variables such as a_{τ_i,τ_j} to real variables, since their values will be enforced to either 0 or 1 by A_{τ_i,e_j}, according to constrains (10) and (11) (A_{τ_i,e_j} is defined as binary variables in the formulation).

As shown in Table II, the total number of variables in the MILP formulation, after the presolve stage in CPLEX, is 34,075, of which 450 are binary variables. The number of constraints is 80,817. A feasible solution satisfying all path deadline constraints was found in 14.5 seconds. The corresponding objective value, that is, the sum of the latencies of given paths, was 33,119.5ms for this feasible solution. The objective value improved with time and after 18,539.9 seconds, the optimal solution for this formulation was obtained by the solver at a value of 12,295.3ms.

In Step 2, we set the solution of Step 1 as the initial configuration and further reduced the objective function by packing the signals and optimizing the priorities assigned to tasks and messages. We used the exact formulation for computing the task and message response times as defined in (1) and (4). There are 3,247 variables in the formulation, of which 1,841 are binary variables and an additional 1,122 are general integer variables. The number of constraints is 18,946. In just 0.45 seconds, the optimal solution with 4,927.5ms total latency was found. The largest latency among all the paths with deadline at 300ms is 111.5ms and the largest latency for 100ms deadline paths is 2.3ms. Since the exact response time calculation was used in the formulation, the latency numbers in our final solution were exact.

5.2. Second Option: Dual Bus Configuration

In the second configuration, the ECUs are connected by two CAN buses, with one ECU functioning as the gateway between the two buses. The transmission speed of both CAN buses is 500kb/s.

Experiment 1: same flow as in the single-bus case. In Step 1, we again used the upper bound formulation (41) for approximating task and message response times. We also relaxed some integer variables to real, while guaranteeing the correctness of their values. The number of variables and constraints are shown in Table III. The optimal

Table III. Result of the Dual-Bus Configuration

Step	#Var	#Int	#Binary	#Con	Time(s)	Result(ms)	Gap
Step 1	32818	0	369	116846	8945.0	15871.5	Opt
Step 2	3598	1375	1934	19231	54.0	4927.8	Opt

Table IV. Result of the Dual-Bus Configuration with Priority Optimization in Step 1

Step	#Var	#Int	#Binary	#Con	Time(s)	Result(ms)	Gap
Step 1	36919	0	2009	189033	50000	4987.5	41.4%
Step 2	3672	1490	1869	16271	38.2	4927.8	Opt

solution for the formulation was found in 8,945.0 seconds with an objective value of 15,871.5ms.

In Step 2, the signal packing, task and message priority assignment are optimized using the exact response time formulation and using the allocation from the previous step. At the end of the optimization, the solver found the optimal solution for this step in 54.0 seconds, with a total latency down to 4,927.8ms. In this final solution, the largest latency is 111.7ms among 300ms deadline paths and 2.3ms among 100ms deadline paths. This is very close to the single bus case since, for those tasks and messages that affect the latencies of selected paths most, their optimal configurations (allocations, priorities, signal packings) are found to be similar in both architecture platforms.

Experiment 2: using exact response time. We conducted an experiment to study the impact of approximating response times in Step 1. In this experiment, after using the upper bounds of response times for task allocation as in Experiment 1, we did not immediately feed the result to Step 2. Instead, we leveraged the 15,871.5ms total latency as a bound for the objective function and repeated the task allocation optimization with the exact response time formulation. The solver found the optimal solution for this exact formulation after 13,576.9 seconds at a total latency of 13,137.4ms. This new result of task allocation is then fed into Step 2. In 42.0 seconds, Step 2 found its optimal solution at 4,927.3ms, which is very close to the 4,927.8ms we got from Experiment 1. This shows that using the upper bound formulation of the response times is a good way to reduce complexity, while maintaining the quality of the results.

We also tried directly running the exact formulation without the bound obtained from the approximated formulation. It took 37,824.4 seconds to find a solution with a cost value of 21,807.9ms. The solution is far from the optimal solution of 13,137.4ms, and also requires a longer runtime (the total time when using the cost bound is $8{,}945.0 + 13{,}576.9 = 22{,}521.9$ seconds).

Experiment 3: optimizing both task allocation and task priority. We conducted another experiment to study the impact of optimizing more design variables in Step 1. We again leveraged the 15,871.5ms upper bound on the total latency obtained from the task allocation optimization, and then synthesized both the task allocation and the priority assignment using the upper bound formulation of the response times. As shown in Table IV, the solver found a solution with an objective value of 4,987.46ms after a time limit of 50,000 seconds. The gap between this solution and the theoretical lower bound estimated by the solver is 41.4%.

Step 2 took the allocation from Step 1 and found the same final solution (with a cost of 4,927.8 ms) as in Experiment 1.

Experiment 4: simulated annealing approach. To evaluate the runtime required by our MILP approach and the quality of the results, we compared our solution with a simulated annealing algorithm (SA) obtained with limited modifications from the one described in Zhu et al. [2009]. We outlined the possible disadvantages of stochastic

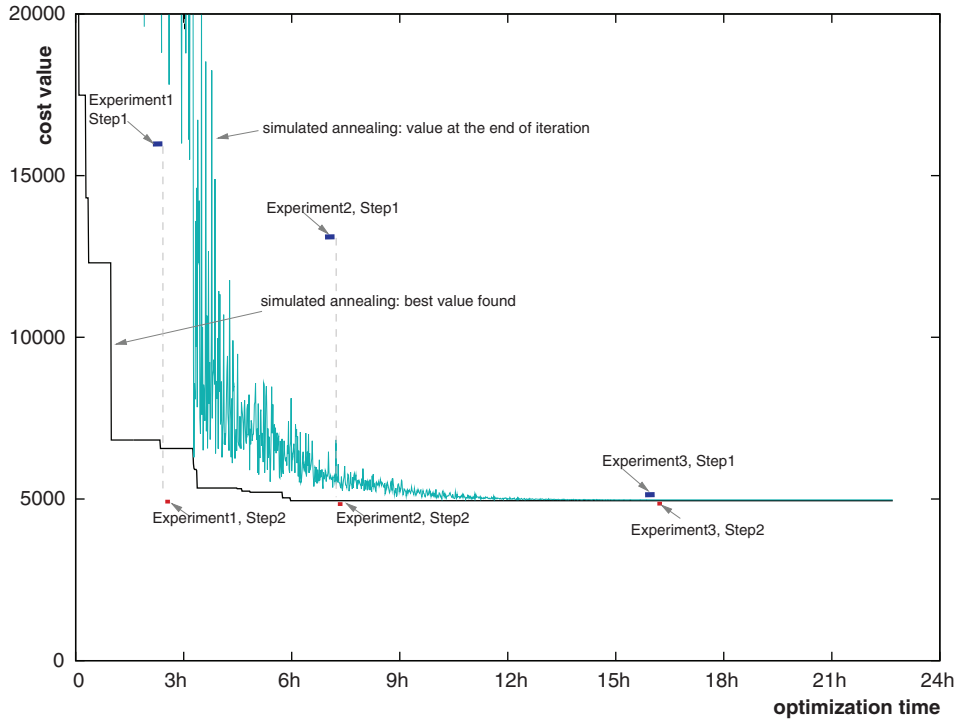

Fig. 6. Optimal value with time for the MILP based approach and Simulated Annealing algorithm.

optimization approaches in Section 1. In addition, in our case study, the SA parameters were very difficult to tune and the algorithm ended up quite often in local optima. The problem is a quite general one and it is worth discussing. In SA, there is no direct way of handling constraints and infeasible intermediate solutions. If it is possible to demonstrate that the solution space is connected (at least for the selected transition operator), a possible approach could be to reject any infeasible intermediate solution. In our case, there is no such guarantee. Therefore, to avoid being stuck in local optima, we conditionally allow infeasible solutions. This presents the problem of how to evaluate such solutions by a suitable modification of the metric function. Clearly, they should be penalized to avoid having an infeasible solution (with a good overall metric value) as the final result. However, a penalty that is too steep could easily result in the impossibility of getting out of an “island" of feasible solutions to reach another region of the solution space containing the global optimum, but not connected by a path of feasible solutions. Of course, this is further complicated by the fact that, if SA is the only available tool, the designer has no indication about the quality of the final result, if it is a local optimum, and how far it is from the global optimum. In our case, we luckily had the results of the MILP approach as a reference and this allowed us to tune the SA metric.

At the end (after several tries), when executed with an initial temperature of 20,000, a cooling rate of 0.99, a final temperature of 0.005, and with a maximum number of iterations at each step equal to 7,000, Simulated Annealing, after almost 23 hours of computation, computed a best value of 4,945.75ms total latency. Figure 6 shows the values of the cost function reached with time when using the different approaches. Two short dashes on the left represent the results for the first experiment (same flow

Table V. Comparison of Different Objective Functions

Experiment		Min Sum Latencies			Min Max Lateness		
		Result(ms)	Time(s)	Gap	Result(ms)	Time(s)	Gap
Single bus	Step 1	11178.3	24900.7	Opt	−93.89	25301.0	Opt
	Step 2	4927.5	0.45	Opt	−97.75	0.32	Opt
dual bus exp. 1	Step 1	15871.5	8945.0	Opt	−94.68	50000	2.61%
	Step 2	4927.8	54.0	Opt	−97.75	87.7	Opt
dual bus exp. 2	Step 1	13137.4	13576.9	Opt	−95.03	25891.0	Opt
	Step 2	4927.3	42.0	Opt	−97.75	7.61	Opt
dual bus exp. 3	Step 1	4987.5	50000	41.4%	−97.60	50000	1.26%
	Step 2	4927.8	38.2	Opt	−97.75	238.23	Opt

as in the single-bus case). The upper plot, in blue color (after 8,945.0 seconds with an objective value of 15,871.5ms) shows the result of the first step. The lower one in red, at almost the same x-coordinate (54 more seconds), with a cost of 4,927.8, is the final outcome for the first experiment. Similarly, two more pairs represent the results and the running times for the first step (in blue) and final step (in red) of Experiments 2 and 3. The two continuous lines represent the result with time of the simulated annealing algorithm. The light line represents the cost of the current solution (which can possibly increase with time, as SA conditionally allows transitions to higher cost solutions), and the black line the best cost found up to some given time. As shown in the figure, the quality of the solutions found in the first step in not very good, but the metric is drastically improved in the second optimization step, which requires a quite short time. For SA, at least 12 hours are needed to achieve a good quality solution, and the final outcome is in any case worse than the results obtained using any of the previous three methods.

5.3. Alternative Objection Function: Minimizing Maximum Lateness

As discussed in Section 3.7, the minimization of the maximum lateness on a set of time-critical paths is a valuable option for a performance function as opposed to our initial choice of the sum of latencies. To show how our formulation easily allows retargetting to a different metric function and what the performance is for this alternative objective, we replaced (70) with (71) in the formulation and applied the same optimization procedures as before. The runtimes and results are summarized in Table V. As a comparison, the results for the original objective are also shown in the table. In general, the results show that the performance of the solver is comparable for both metrics, with the optimal solution reached in many cases (in others with a small gap or short distance from optimality). For some cases, the application of the new metric results in shorter processing times, for others in longer times, but always within acceptable values, showing the generality of the proposed optimization procedure with respect to different objective functions. The difference in the runtimes for the two functions is very difficult to explain in detail for the individual cases. However, we can explain what the trade-offs are that are the likely cause of this behavior. In the sum-of-latencies metric, there is a single metric function and all path latencies need to be considered in the optimization stage. In the min-of-max lateness case, the encoding of the function requires a larger number of constraints. Also, low-latency paths clearly do not contribute in this case and are easily pruned, but the solver can spend significant time in optimizing the few critical paths having to resolve conflicting configurations (that would decrease the latency of one path but increase it on another). In contrast, the sum-of latencies function tends to be a characterization of a system average behavior, and configurations that

Table VI. Scalability Analysis of the Dual-Bus configuration

#tasks	Step	#Var	#Int	#Binary	#Con	Time(s)	Result(ms)	Gap
41	Step 1	32654	0	369	116685	68534	−97.43	Opt
	Step 2	7435	2990	4283	71355	24.5	−98.88	Opt
46	Step 1	39035	0	414	137279	20231.8	−97.43	Opt
	Step 2	11403	4452	6701	88431	129.55	−98.88	Opt
51	Step 1	51775	0	459	183982	21430.1	−97.43	Opt
	Step 2	13876	6112	7520	125121	12458.42	−98.88	Opt
56	Step 1	62984	0	504	223333	15573.7	−97.43	Opt
	Step 2	16796	7293	9192	144658	1108.6	−98.88	Opt
61	Step 1	75294	0	549	266483	50000	−96.14	1.35%
	Step 2	28357	12038	15970	400885	75000	N.A.	N.A.

expose conflicts among paths tend to be smoothed out (the reduction on one path may compensate the increase on another).

In this case, retargetting the simulated annealing algorithm to the new metric proved to be quite easy (the new metric accommodates additional penalties for infeasible solutions in a much easier way). The simulated annealing procedure found the same best solution as the MILP approach at −97.75 (for the dual-bus configuration), with similar runtimes as in the previous case.

5.4. Applicability and Scalability

Our case study does not answer a very interesting question, that is: "what is the maximum problem size that the method can handle?". There are so many dimensions to our problem that the definition of the region of applicability is practically impossible. The number of ECUs and buses, the system topology, the computation speeds of ECUs and the transmission speed of buses, the number of tasks and their periods and computation times, the number of signals and the topology of the functional communication, and finally the size of signals can all come into play. Exploring all of these dimensions exhaustively or even a significant subset is impractical, especially considering that, for problems of industrial size, optimization times of several hours are expected.

Nevertheless, to give some hints about the increase of running time with increasing problem size (without any pretense at obtaining a boundary for applicability), we tried optimizing a series of configurations in which the functional architectures (tasks and signals) of our case study is gradually increased and mapped into the same architecture of 9 ECUs and 2 buses. We started from our original architecture and artificially extended it by duplicating a subset of its tasks and signals. We increased the task set size by 5 tasks at a time, including the signals exchanged among them. In order to avoid running the risk of generating infeasible configurations, we also halved the computation times of all tasks. The metric function considered for this study is the minimization of the maximum lateness. The sizes of the problems and their respective optimization times and results are summarized in Table VI.

Optimal solutions can be found for all but the last application configuration (of 61 tasks), for which we could not find a solution to Step 2 in almost 21 hours of runtime. Please note how the runtime is not a monotonic function of the number of tasks. In fact, a longer optimization time may be needed for a smaller set (with lower utilization). This can be the result of a lower utilization which allows a larger number of solutions to be feasible and causes the solver to spend more time to explore all of them. For example, in the case of 41 tasks, a solution with a gap of 1.15% was found in a short time of only 136 seconds. This solution is indeed the optimal solution. However, the solver needed more than 68,000 seconds (approx 20 hours) to prove its optimality,

because of the large number of feasible solutions. However, when we add more tasks and signals, the utilization becomes higher and the time may again rise as in the case of 61 tasks. This is because of the sheer complexity (larger number of tasks and signals) and also the fact that the solver spends lots of time looking for feasible solutions.

We truly believe that for this class of problems scalability simply cannot be defined by (brute force) experiments. While more cases would definitely be useful, considering that the analysis of a single case takes approximately one day of computation time, the coverage of the possible system configurations would still be extremely small. In conclusion, for systems of several tens of tasks, the designer would still need to try his/her own system configuration. For example, our case study, as many of today's automotive systems, is characterized by high connectivity among tasks (more than 170 paths for 10 endpoints). Other systems may exhibit a different communication pattern. We can add that, based on our experience, there is often a discontinuity in running times and gap size. At some point, the runtime of the solver suddenly increases. Also, the gap remains very small for some problems sizes but then suddenly increases to very large values (consider e.g., Step 2 for 61 tasks, which could not be solved, even for a suboptimal feasible solution, within the timeout).

6. CONCLUSIONS

This article presented an integrated formulation for optimizing task allocation, signal packing, as well as task and message priorities to meet end-to-end deadline constraints and minimize latencies in distributed automotive systems. To make it practical for industrial-size applications, a two-step synthesis method approximating the integrated formulation was proposed to reduce the complexity. We applied this method to an automotive case study, and showed it can effectively reduce the end-to-end latencies. Although the method targets only a part of the architecture definition stages, it is flexible enough to accommodate many constraints of interest, and can effectively help the designer in solving the problems of practical size. The proposed technique should be merged with optimization results already devised for the assignment of the periods and the task and messages activation modes as detailed in previous reports.

REFERENCES

Aström, K. J. and Wittenmark, B. 1990. *Computer-Controlled Systems: Theory and Design* 2nd ed. Prentice-Hall, Inc., Upper Saddle River, NJ.

AUTOSAR. 2010a. Autosar release 4.0 os specifications. http://www.autosar.org/download/R4.0/5AUTOSAR_SWS_OS.pdf.

AUTOSAR. 2010b. Autosar release 4.0 specifications. http://www.autosar.org/.

Bate, I. and Emberson, P. 2006. Incorporating scenarios and heuristics to improve flexibility in real-time embedded systems. In *Proceedings of the 12th IEEE RTAS Conference*. 221–230.

Benveniste, A., Caspi, P., Edwards, S., Halbwachs, N., Guernic, P. L., and de Simone, R. 2003. The synchronous languages 12 years later. *Proc. IEEE 91*.

Bini, E. and Buttazzo, G. C. 2002. The space of rate monotonic schedulability. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium*. 169–178.

Bini, E., Huyen Châu Nguyen, T., Richard, P., and Baruah, S. K. 2009. A response-time bound in fixed-priority scheduling with arbitrary deadlines. *IEEE Trans. Comput. 58,* 2, 279–286.

Bini, E., Natale, M. D., and Buttazzo, G. 2006. Sensitivity analysis for fixed-priority real-time systems. In *Proceedings of the Euromicro Conference on Real-Time Systems*.

Bosch, R. 1991. Can specification, version 2.0.

Boyd, S., Kim, S., Vandenberghe, L., and Hassibi, A. 2006. A tutorial on geometric programming. *Optimiz. Engin.* To appear.

Brook, D. 2000. Embedded real time operating systems and the osek standard. In *Proceedings of the SAE World Congress.*

Casparsson, L., Rajnak, A., Tindell, K., and Malmberg, P. 1999. Volcano, a revolution in on-board communications. Tech. rep. Volvo Technology.

Davis, R. I. and Burns, A. 2009. Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. In *Proceedings of the 30th IEEE Real-Time Systems Symposium*. 398–409.

Davis, R. I., Burns, A., Bril, R. J., and Lukkien, J. J. 2007. Controller area network (can) schedulability analysis: Refuted, revisited and revised. *Real-Time Syst. 35,* 3, 239–272.

Flexray. 2006. Protocol specification v2.1 rev. a. http://www.flexray.com.

Gerber, R., Hong, S., and Saksena, M. 1995. Guaranteeing real-time requirements with resource-based calibration of periodic processes. *IEEE Trans. Soft. Engin. 21,* 7, 579–592.

Gutiérrez, J. P., García, J. J. G., and Harbour, M. G. 1997. On the schedulability analysis for distributed hard real-time systems. In *Proceedings of 9th Euromicro Workshop on Real-Time Systems*. 136–143.

Hamann, A., Henia, R., Jerzak, M., Racu, R., Richter, K., and Ernst, R. 2004. SymTA/S symbolic timing analysis for systems. http://www.symta.org.

Hamann, A., Racu, R., and Ernst, R. 2006. A formal approach to robustness maximization of complex heterogeneous embedded systems. In *Proceedings of the CODES/ISSS Conference*.

Hamann, A., Racu, R., and Ernst, R. 2007. Multi-dimensional robustness optimization in heterogeneous distributed embedded systems. In *Proceedings of the 13th IEEE RTAS Conference*.

Harbour, M. G., Klein, M., and Lehoczky, J. 1994. Timing analysis for fixed-priority scheduling of hard real-time systems. *IEEE Trans. Softw. Engin. 20,* 1.

He, X., Gu, Z., and Zhu, Y. 2009. Task allocation and optimization of distributed embedded systems with simulated annealing and geometric programming. *Computer J.*

Henriksson, D., Cervin, A., and Årzén, K.-E. 2002. Truetime: Simulation of control loops under shared computer resources. In *Proceedings of the 15th IFAC World Congr. Automatic Control*.

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

Kienhuis, B., Deprettere, E. F., van der Wolf, P., and Vissers, K. A. 2002. A methodology to design programmable embedded systems - the Y-chart approach. In *Embedded Processor Design Challenges: Systems, Architectures, Modeling, and Simulation-(SAMOS)*, E. F. Deprettere, J. Teich, and S. Vassiliadis, Eds., Lecture Notes in Computer Science, vol. 2268, Springer, 18–37.

Kopetz, H., Damm, A., Koza, C., Mulazzani, M., Schwabla, W., Senft, C., and Zainlinger, R. 1989. Distributed fault-tolerant real-time systems: The mars approach. *IEEE Micro 9,* 1.

Lakshmanan, K., Niz, D. d., and Rajkumar, R. 2009. Coordinated task scheduling, allocation and synchronization on multiprocessors. In *Proceedings of the 30th IEEE Real-Time Systems Symposium*. 469–478.

Lehoczky, J. P. 1990. Fixed priority scheduling of periodic task sets with arbitrary deadline. In *Proceedings of the 11th IEEE Real-Time Systems Symposium*. 201–209.

Liu, C. L. and Layland, J. W. 1973. Scheduling algorithms for multiprogramming in a hard real-time environment. *J. ACM 20,* 1, 46–61.

Liu, J. W. S. 2000. *Real-Time Systems*. Prentice Hall.

Metzner, A. and Herde, C. 2006. Rtsat– an optimal and efficient approach to the task allocation problem in distributed architectures. In *Proceedings of the 27th IEEE International Real-Time Systems Symposium*. IEEE Computer Society, Los Alamitos, CA, 147–158.

OSEK. 2006. Osek os version 2.2.3 specification. http://www.osek-vdx.org.

Padmanabhan, K. Z., Pillai, P., and Shin, K. G. 1999. Emeralds-osek: A small real-time operating system for automotive control and monitoring. In *Proceedings of the SAE International Congress and Exhibition*.

Palencia, J. C. and Harbour, M. G. 1998. Schedulability analysis for tasks with static and dynamic offsets. In *Proceedings of the 19th IEEE RTSS Conference*. 26.

Palencia, J. C. and Harbour, M. G. 1999. Exploiting precedence relations in the schedulability analysis of distributed real-time systems. In *Proceedings of the 20th IEEE Real-Time Systems Symposium*. 328.

Pop, T., Eles, P., and Peng, Z. 2002. Holistic scheduling and analysis of mixed time/event-triggered distributed embedded systems. In *Proceedings of the 10th International Symposium on Hardware/Software Codesign*. 187–192.

Pop, T., Eles, P., and Peng, Z. 2003. Design optimization of mixed time/event-triggered distributed embedded systems. In *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. ACM Press, New York, NY, 83–89.

Racu, R., Jersak, M., and Ernst, R. 2005. Applying sensitivity analysis in real-time distributed systems. In *Proceedings of the 11th Real Time and Embedded Technology and Applications Symposium*. 160–169.

RAMAMRITHAM, K., FOHLER, G., AND ADAN, J. M. 1993. Issues in the static allocation and scheduling of complex periodic tasks. In *Proceedings of the 10th IEEE Workshop on Real-Time Operating Systems and Software*. IEEE Computer Society, 11–16.

SAKET, R. AND NAVET, N. 2006. Frame packing algorithms for automotive applications. *J. Embed. Comput. 2,* 1, 93–102.

SAKSENA, M. AND HONG, S. 1996. Resource conscious design of distributed real-time systems – an end-to-end approach. In *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems*.

SANDSTROM, K., NORSTOM, C., AND AHLMARK, M. 2000. Frame packing in real-time communication. *Proceedings of the 7th International Conference on Real-Time Computing Systems and Applications*. 399–403.

SHA, L., RAJKUMAR, R., AND LEHOCZKY, J. P. 1990. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Trans. Comput. 39,* 9.

TINDELL, K. W. 1993. Holistic schedulability analysis for distributed hard real-time systems. Tech. rep. YCS 197, Department of Computer Science, University of York.

ZHENG, W., ZHU, Q., NATALE, M. D., AND VINCENTELLI, A. S. 2007. Definition of task allocation and priority assignment in hard real-time distributed systems. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium*. 161–170.

ZHU, Q., YANG, Y., SCHOLTE, E., NATALE, M. D., AND SANGIOVANNI-VINCENTELLI, A. 2009. Optimizing extensibility in hard real-time distributed systems. In *Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium*. 275–284.

Received February 2010; revised August 2010; accepted November 2010