International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 5 (2018) pp. 2368-2374

Scheduling Analysis and correction for Multiprocessor Real-Time Systems Based on Multi-Agent Systems

Adel Mahfoudhi[1,2], Walid Karamti[3] and Atef Zaguia[1]

[1]College of Computers and Information Technology, Taif University, Taif, Saudi Arabia.
[2] Computer and Embedded Systems Laboratory, ENIS, University Of Sfax, Tunisia.
a.mahfoudhi@tu.edu.sa
[3]Higher Institute of Computer Science and Mathematics, University of Monastir, Tunisia.

Abstract

The Partitioned multiprocessor scheduling is the most used in practice for multiprocessor Real-Time Systems (RTS). Its complexity is known as NP-Hard and it is very time consuming to find a schedulable partition.

Hence, the only existing solution to correct a non-schedulable partition is the regeneration of a new one.

The present paper presents a new approach for scheduling analysis and correction in the aim of accelerating the process of finding a schedulable solution. In fact, we have used the multi-agent systems and the contract net protocol for modeling a net of cooperating processors able to correct, if possible, a non-schedulable solution without regeneration.

Keywords: Real-Time Systems, Multi-Agent Systems, Contract Net, Scheduling analysis and correction

INTRODUCTION

The he Real-Time systems (RTS) are omnipresent in several domains such as the control of nuclear power stations, multimedia communications, robotics, avionics, system on chip (SOC) [23, 26, 2], etc. Therefore, the RTS are characterized by complex applications that require powerful architectures to be satisfied. The architecture can be specified with a powerful processor or, for equal power, a set of low processors. In practice, a multiprocessor architecture composed of low processors is required due to two major facts. First, a multiprocessor architecture is much cheaper compared to single-processor architecture. Second, it is crucial to distribute the calculus on various specialized processors for some domains like the SOC.

The multiprocessor scheduling presents a recent research area whose results are still in progress. Two scheduling families can be scheduled, the first of which is the global scheduling which is characterized by the liberty of all the tasks to continue their execution in any free processor. However, the cost of migration and preemption is so important [4, 13, 19]. As for the second one, it is the partitioned family that is based on the reduction of the preemption and the migration costs [16]. In fact, the tasks are distributed over the processors during the assignation phase. Each partition is considered as a single-processor scheduling problem in which the optimal scheduling policy exists [14]. Then, a scheduling analysis of all partitions is made to assure that the RTS system is safe.

The partitioned multiprocessor scheduling is used in a several RTS applications, such as the multimedia applications, the avionic application etc. It aims to use all the existing processors to reach a higher possible performance [3, 20].

A. Classical Partitioned Multiprocessor scheduling strategy

The classical partitioned multiprocessor scheduling strategy is able to schedule the RTS application on the RTS architecture (Figure 1). In fact, the RTS application is described with a finite set of tasks and a precedence relation between all the tasks.

The schedule is based on two major steps: the partition of tasks corresponding to each processor and checking the schedulabilty of each one (Figure 1).

If each partition is schedulable, then the deployment of the application on the architecture can be carried out safely, otherwise to correct it, a new partitioning is recommended.

The partitioning consists in distributing 'n' set of tasks into 'm' parts and attributing each set to a processor. In fact, each partition tasks is not allowed to migrate from its initial partition towards another during the scheduling.

Each processor is defined with a capacity to execute a finite number of tasks. In fact, each task can be defined with its necessary duration requested to be executed on the processor.

Respecting this definition, the scheduling problem is known as a classification problem. Indeed, it is a problem of bin packing to distribute packets (tasks) in boxes (processors) [6].

In the literature, many optimal algorithms have been used to solve the bin packing NP-hard problem [10, 8]. In fact, Coffman and Csirik have made a good presentation for the one-dimensional bin- packing problem in [11]. However, these algorithms are effective only for a limited number of elements [5]. Indeed, for such NP-hard problems, the most common is the use of "heuristics" [8].

Among the main heuristics used, the simple algorithms used are first fit decreasing (FFD) and best fit decreasing (BFD). Several of the latter have been designed for different application domains to find approximate solutions when there is no exact method or when the solution is unknown

[18].However, these algorithms are non-optimal. In fact, they generally allow obtaining results close to the optimal solution [7].

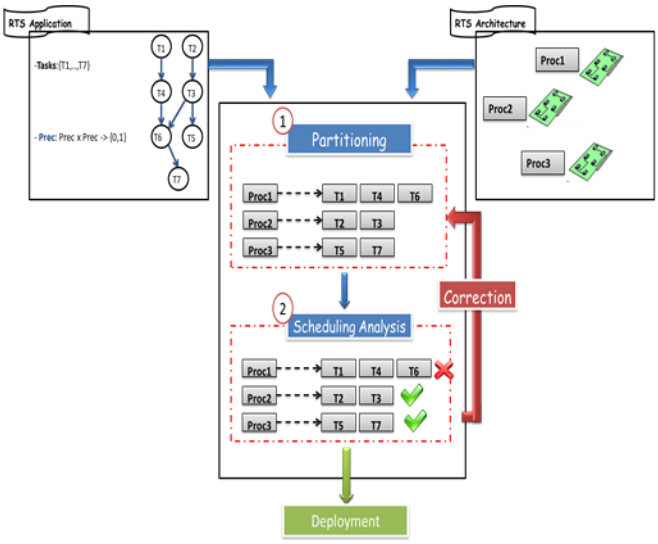

Figure 1: Classical Partitioned scheduling approach

Each partition is considered as a single-processor scheduling problem in which the optimal scheduling policy exists [14]. Then, a scheduling analysis of all the partition is made to assure that the RTS system is safe.

In literature, the scheduling analysis is tackled with various techniques in order to protect the RTS from failures. Particularly, the formal method presents the most secure technique proposed for the scheduling analysis.

Generally, the choice of the suitable formal method depends on the characteristics of the system and the properties to be checked. We distinguish two main classes from prove methods for scheduling analysis which analytical methods and model checking methods [17].

Contrary to the model checking methods, the analytical verification provides algorithms characterized by polynomial complexity for schedulability checking of a set of tasks [15]. In fact, the results of the analysis are provided very quickly compared to other methods. This can explain why this method is very used in practice.

Unfortunately, despite the amount of research in this field, there are still common problems that always occur in the proposed scheduling partitioning solutions. The most common is the inability to give a reliable partitioning solution from the beginning, which increases the risk of non-schedulability. In fact, in order to correct the schedulability, the partitioner is called for a new regeneration from an exponential number of partitioning possible solutions (Figure 1). Indeed, the correction of non-schedulable partitions is a costly task. Hence, it is very interesting to try the correction of the considered solution to reduce the increasing cost of regeneration.

B. Contribution and outline of the paper

The main contribution of this paper is the proposition of a new approach of scheduling analysis and correction into the multiprocessor partitioned scheduling in order to decrease the time cost to find schedulable partitions. The solution is based on Multi-Agent systems and the Contract Net protocol in the aim to create a net of processors. Indeed, the proposed solution allows a processor with non-schedulable tasks to cooperate with others to find a new allocation to each one.

The rest of the paper is as follows: we present in section II the proposed scheduling analysis and correction. In section III, we present the proposed Multi-Agent model for the correction. Next, section IV details a case study and provides the ability of the proposed approach to correct all the described non-schedulable tasks. Finally, we summarize and provide future research directions, in section V.

PROPOSED SCHEDULING ANALYSIS AND CORRECTION APPROACH

In order to explain our approach, we start with the description of the used RTS, and then we define the method to analyze and correct a given partitioning solution.

C. RTS Model

The used RTS is specified with system Ω. It is defined by the 4-tuplet:

$$\Omega =< \text{Task}, \text{Proc}, \text{Alloc}, \text{Prec} >$$

with:

• Task : {T1, T2,..., Tn}, with (n ≥ 0) is the number of tasks; each Ti ∈ Task is determined by Ti = <Ri , Pi , Ci> where,

- Ri , is the date of the first activation,
- Pi , the period associated with the task,
- Ci , the execution period of the task for the Pi period.

• Proc : {P1, P2,..., Pm}, with m ≥ 0 is the number of processors.

• Alloc : Task → Proc, a function which allocates a task to a processor. Alloc is a surjective function. In fact, a processor is allocated to at least one task, but a task must be assigned to only one processor.

• Prec : Task × Task →{0, 1}, a function which initializes precedence relations between tasks.

We consider in this paper only the case of independent tasks $\forall T_i, T_j \in \text{Task}, \text{Prec}(T_i, T_j) = 0$ with simultaneously start $(R_i = 0)$.

The utilization factor of task Ti is denoted by ui where ui = Ci/Pi, and the capacity of a processor Procj by:

$$U_j = \sum_{i=1}^{k} \left({}^{C_i}/_{P_i} \right)$$

With k is the number of tasks composing the partition of the processor Procj. A partitioned multiprocessor RTS is schedulable if all the processors do not exceed their corresponding capacities. The scheduling analysis techniques are able to identify all the non-schedulable partitions without proposing a solution for correction.

D. Proposed approach

The objective of the proposed approach (Figure 2) is to decrease the time to define a feasible partitioning. To do so, starting with a given partitioning solution, we are based on two facts: using a fast scheduling analysis and correcting the non-schedulable partitions without the regeneration of a new configuration.

In order to accelerate the scheduling analysis, we used an analytical method presented in [21]. The results of this step are principally two kinds of partitions: schedulable and non-schedulable. If the processor's partitions cooperate then the correction can be established. In fact, in a feasible RTS, the existing of non-schedulable partitions indicates that some schedulable partitions are relaxed and able to receive more tasks for execution.

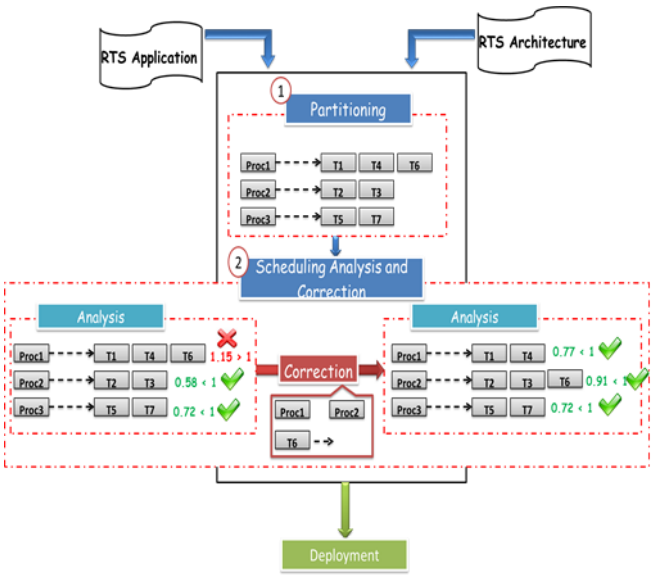

Figure 2: Proposed Scheduling analysis and correction

Regarding the processors capacities, we can define schedulable and non-schedulable partitions as follows:

- Schedulable partition: when the capacity is inferior to 1. All the tasks of this partition are schedulable and there is a free space to schedule other ones.

- Non-schedulable partition: when the capacity is superior to 1. Some tasks have no places on the processor to be executed. So, if they migrate to the schedulable partitions, then the RTS can be corrected without the regeneration of a new partitioning solution.

In practice, we may have a case with the capacity equal to 1, that is when all the tasks of this partition are schedulable and no free space to receive others. We called this kind of partition a Blocked partition.

II. Multi-Agent Systems for the Scheduling Analysis and Correction

In order to correct the non-schedulable partitions, it is so interesting to find a way to let all the partitions communicate, cooperate and exchange tasks. In fact, the non-schedulable partition asks the schedulable ones for places to receive tasks. This can be done if we model each of processor with an autonomous agent [9] and if we consider an appropriate communication mechanism.

A. Overview

To fit the needs of different domains, various interaction protocols have been developed for task allocation in multi-agent systems. Some of them are built for specific domains, like the monotonic concession protocol and the corresponding Zeuthen strategy [12] for task oriented domains where agents have to reallocate a set of tasks. Such protocols cannot be used in more general situations. Other mechanisms are designed for more general situations where an agent needs to allocate some task to the most suitable agent. Examples of such protocols are different kinds of auctions (English auction, Dutch auction and first-price sealed bid auction). One of the most popular used protocols [25, 24] is the Contract NET interaction protocol specified as an interaction protocol by the Foundation of Intelligent Physical Agents (FIPA) [1] and used in practical applications.

The Contract NET protocol has several advantages over other protocols. Firstly, it allows finding an agent that is the most suitable for the task. Secondly, it is the only protocol that has been accepted by FIPA as a standard, and no longer has experimental status [1]. Thus, it is standardized, widely used and well known to the developers of multi-agent systems. Additionally, it is reliable in the sense that if an individual agent becomes unavailable, the task can be easily reassigned to another agent [22]. We consider the autonomous agent with Contract NET to model the scheduling analysis and correction approach.

B. Proposed Model

The Contract Net appeared for the first time in [22]. It is characterized with two types of agents: Initiator and Participant. Indeed, at any time, any one agent can be an At an initiator, participant or both. Besides, it allows contracting as well as subcontracting.

This protocol aims to execute a sequence composed by 3 steps. First, the initiator agent sends out a call for proposals. Second, each Participant reviews the proposal and bids on feasible ones. Finally, the initiator chooses the best bid, awards a contract to that participant and rejects other bids.

Based on this description, we propose our model for the scheduling analysis and correction (Figure 3).

The proposed model is composed of two types of agents: allocator agent and processor agent. In fact, the processor agent presents one processor and it is responsible to analyze the schedulability of its partition. The allocator agent presents a manager able to manage the communication between the processors agents and search a new location of the non-schedulable tasks. Both of agents are an initiator and participant agent.

The processor agent analyzes its partition based on priority scheduling analysis ([21]). If the analysis fails and the partition is non-schedulable, the correction process is then started and it is based on two major steps. First, the local correction, the agent relaxes the processor by selecting some task(s) to be excluded from its partition. Second, the agent allocator is called to search a new allocation to those task(s).

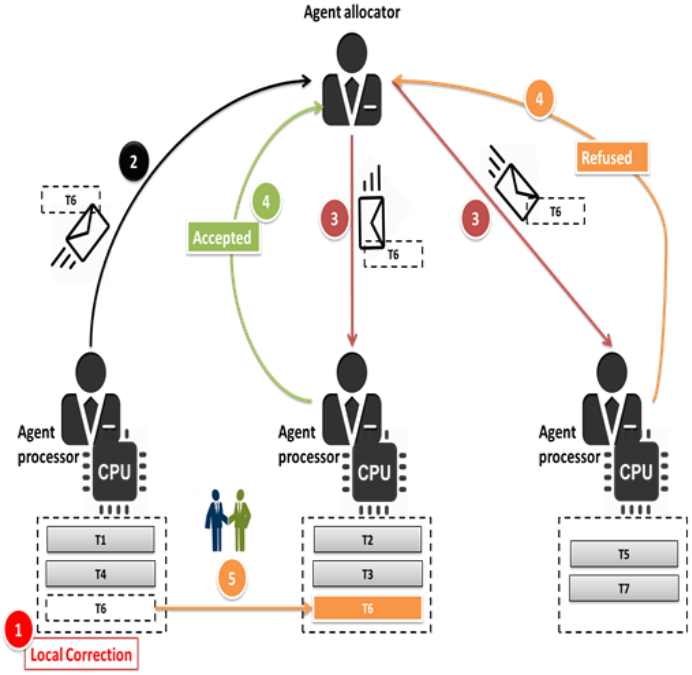

Figure 3: Scheduling analysis and correction model

The agent allocator receives a set of tasks for re-allocation from the processors agents with a failed analysis. Then, for each task, the allocator agent sends a call to the rest of processor agents of the proposition, including more tasks in their partition. Thus, the allocator begins with tasks with the highest utilization factor, because finding a new allocation is more complicating then the lowest. The processor agent can respond with two possible requests: refuse (if it is a blocked partition or non-sufficient space) or accept the proposal.

When all the proposals are comeback, the agent allocator chooses the best partition based on the Best-Fit algorithm, and then, the task will be included in the chosen partition.

This process will be iterated until the re-allocation of all the tasks.

CASE STUDY

In the present section, we introduce a demonstration of the proposed scheduling and correction approach through a case study.

In fact, we present a generic experiment that consists of a non-schedulable system. In the latter, we establish the way how the proposed approach supplies a description of the non-schedulable tasks and how to find a new allocation to each one in order to correct the system.

The case study deals with 15 independent tasks running on 4 identical processors. Using definition 1, the specifications of the task characteristics as well as the allocation of the processors by the tasks are described as follows.

Task = {T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15}

Proc = {P1, P2, P3, P4}

Table 1: Task characteristics and allocation

Task	Ri	Pi	Ci	Allocation
T1	0	16	3	P1
T2	0	3	1	P2
T3	0	8	1	P1
T4	0	15	8	P3
T5	0	7	2	P4
T6	0	9	4	P2
T7	0	8	3	P1
T8	0	7	3	P4
T9	0	16	2	P1
T10	0	9	2	P2
T11	0	7	2	P4
T12	0	7	1	P4
T13	0	15	3	P3
T14	0	9	1	P2
T15	0	15	1	P3

Four partitions are created; with each one an agent is associated. The scheduling analysis results indicate two schedulable partitions (P1, P3) and two non-schedulable (P2, P4) (step 0, table 2). Thus, the correction process is triggered.

In order to describe all the correction steps, we present in the Table 2, the interaction between the agents and all the established contracts and deals.

Table 2: Description of the correction steps

Step	*Description*
0	Capacity of processors:U1=0.8125 U2=1.1111 U3=0.8000 U4=1.1428
1	Agent_P2: T14 is non-schedulable Agent_P4: T12 is non-schedulable
2	Agent_P2 calls the Agent allocator to reallocate T14 (u14 = 0.1111) Agent_P4 calls the Agent allocator to reallocate T12 (u12 = 0.1428)
3	Agent allocator selects the task (T12) and sends it as a proposal to the Agent P1 and Agent P3.
4	Agent_P1 accepts the proposal to schedule T12, the new U1=0.9553 Agent_P3 accepts the proposal to schedule T12, the new U3=0.9428.
5	Agent allocator deals with the Agent P1 Agent_allocator rejects the proposal of the Agent_P3
6	Agent_allocator deals with the Agent_P4 Capacity of processors: U1=0.9553 U2=1.1111 U3=0.8000 U4=1.0000
7	Agent_allocator selects the task (T14) and sends it as a proposal to the Agent_P1, Agent_P3 and Agent_P4.
8	Agent_P1 rejects the proposal to schedule T14 Agent_P3 accepts the proposal to schedule T14, the new U3=0.9111 Agent_P4 rejects the proposal to schedule T14
9	Agent allocator deals with the Agent_P3
10	Agent allocator deals with the Agent_P2 Capacity of processors: U1=0.9553 U2=1.0000 U3=0.9111 U4=1.0000

The first action into the correction process is the local correction. In fact, each one of the Agent P2 and Agent P2 calculates the utilization factor able to relax the processor to schedule its partition correctly. The Agent P2 aims to reallocate a space ≤ 0.1111. Thus, it corresponds to the factor utilization of the task T14, so, a reallocation of T14 can solve this problem (i.e. task T12 for the Agent P4.) (step 1).

In the next step, step 2, the Agent P2 and Agent P2 call the Agent allocator to find a new allocation of the tasks: T12 and T14.

Before accepting or refusing the proposals, the Agent allocator collects each coming non-schedulable task and starts with the biggest one to find a new allocation for it. If the search is successful then it deals with the origin agent, otherwise it declines the proposal and no contract will be done. Step 3 describes the choice of the task T12 and the call for the other agents' processor to find a new allocation for T12.

The agent calculates the capacity before responding to the calls. If the capacity is not exceeded after including the task, then it accepts then the proposal and it indicates the new

capacity, otherwise it refuses the proposal. In the case when more than one agent accept the proposal, the Agent allocator deals with the agent with the best proposal, based on the Best Fit policy, and declines the others.

When the new allocation is found, the Agent allocator deals with the original Agent processor (deal with Agent P4, step 6). Indeed, when no new allocation is found, the Agent allocator declines the proposal and the system is declared as non-correctable.

After the correction of a partition, it can receive, if its capacity is not blocked, more tasks from other non-schedulable partitions (step 7).

The correction is terminated when all non-schedulable tasks are reallocated.

CONCLUSION

Partitioned multiprocessor Real-Time scheduling has been used more often than the Global scheduling, and are widely supported by commercial real-time operating systems.

However, the partitioning of tasks over the processors is known to be an NP-hard problem (bin-packing problem) in the strong sense. Thus, it is very difficult to find a schedulable solution in polynomial time.

In this paper, we are interested to correct the proposed partitioning given by a partitioner tool without a new regeneration of a new partitioning solution.

To this end, we have proposed a new approach of scheduling and correction approach into the RTS partitioned scheduling. In fact, we have proposed a Multi-agent model for the correction. Each partition is modeled with an agent able to analyze and correct its partition locally with excluding non-schedulable tasks. Besides, we have proposed a supervisor agent able to search a new allocation for the non-schedulable tasks. The communication between the agents is based on the Contract Net protocol.

We have considered in this work an RTS composed of periodic independent tasks and we have used a case study to explain how our approach can be used for the analysis and correction. However, in real applications, tasks are dependent

(Precedence relation and shared resource). Therefore, it is interesting to consider in future works a real-world application and demonstrate the advantage of our approach.

The correction may occur only if a partition presents an adequate space to receive the whole non-schedulable tasks. However, this is one of the major limits of our approach. In fact, we have met some cases that the Agent allocator declines the proposal of correction because no space is able to receive the task, but, if the task is divided on two partitions it can then be corrected. Thus, it is very interesting to improve our approach to be able to divide tasks to be executed on multiple processors (even multi-cores).

REFERENCES

[1] FIPA Interaction Protocol Specifications. http://www.fipa.org/repository/ips.php3, accessed October 2017.

[2] H. Ahn, H. Oh, and Y. Chung. A semi-real-time scheduler for service robot components on windows nt systems. Information: An International Interdisciplinary Journal, 14(5):1629–1644, 05 2011.

[3] T. P. Baker. A comparison of global and partitioned edf schedulability tests for multiprocessors. Technical report, In International Conf. on Real-Time and Network Systems, 2005.

[4] M. Bertogna and S. K. Baruah. Tests for global edf schedulability analysis. Journal of Systems Architecture - Embedded Systems Design, 57(5):487–497, 2011.

[5] M. Chéramy, P-E. Hladik, and A.M. D´eplanche. Real-time scheduling algorithms for multiprocessor. Journal Européen des Syst`emes Automatisés (JESA), 48(7-8):613–639, October 2015.

[6] A-M. Déplanche. Ordonnancement temps réel multiprocesseur: panorama sur les politiques globales. In Ecole d'été temps réel, 2011.

[7] F. Dorin, P. M. Yomsi, J. Goossens, and P. Richard. Semi-partitioned hard real-time scheduling with restricted migrations upon identical multiprocessor platforms. CoRR, abs/1006.2637, 2010.

[8] M. R. Garey and D. S. Johnson. Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA, 1990.

[9] Z. Guessoum and J-P. Briot. From active objects to autonomous agents. IEEE Concurrency, 7(3):68–76, 1999.

[10] D.S. Johnson. Fast algorithms for bin packing. J. Comput. Syst. Sci., 8(3):272–314, June 1974.

[11] E.G. Coffman Jr. and J. Csirik. Performance guarantees for one dimensional bin packing. In Handbook of Approximation Algorithms and Metaheuristics. 2007.

[12] J.Zlotkin. Rules of Encounter - Designing Conventions for Automated Negotiation among Computers. MIT Press, 1994.

[13] S. Kato and N. Yamasaki. Global edf-based scheduling with laxity-driven priority promotion. Journal of Systems Architecture - Embedded Systems Design, 57(5):498–517, 2011.

[14] S. H. Kwang and J.Y.-T. Leung. On-line scheduling of real-time tasks. In IEEE Real-Time Systems Symposium, pages 244–250, 1988.

[15] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. J. ACM, 20:46–61, January 1973.

[16] L.Sha, T. Abdelzaher, K.E. arz´en, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and K.A. Mok. Real time scheduling theory: A historical perspective. Real-Time Systems, 28:101–155, 2004.

[17] A. Mahfoudhi, Y. Hadj Kacem, W. Karamti, and M. Abid. Compositional specification of real time embedded systems by priority time petri nets. The Journal of Supercomputing, 59(3):1478–1503, 2012.

[18] M. P. Panos. Genetic algorithms & engineering optimization by mitsuo gen; runwei cheng. 44:739–740, 01 2002.

[19] M. Park, S. Han, H. Kim, S. Cho, and Y. Cho. Zl scheme: Generalization of edzl scheduling algorithm for real-time multiprocessor systems. Information: An International Interdisciplinary Journal, 8(5):683–691, 09 2005.

[20] J.A. Stankovic S. Cheng and K. Ramamritham. Scheduling algorithms for hard real-time systems–a brief survey. Technical report, Amherst, MA, USA, 1987.

[21] L. Sha, M. H. Klein, and J. B. Goodenough. Rate Monotonic Analysis for Real-Time Systems, pages 129–155. Springer US, Boston, MA, 1991.

[22] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Trans. Computers, 29(12):1104–1113, 1980.

[23] F. Trivino, J.L. S´anchez, F. J. Alfaro, and J. Flich. Network-on-chip virtualization in chip-multiprocessor systems. Journal of Systems Architecture - Embedded Systems Design, 58(3-4):126–139, 2012.

[24] J. Vokrınek, J. Bıba, J. Hodık, J. Vybıhal, and M. Pechoucek. Competitive Contract Net Protocol, pages 656–668. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[25] M. Wooldridge. An Introduction to Multi Agent Systems. Wiley Publishing, 2nd edition, 2009.

[26] X. Ye, L. Zhu, and S. Guo. Deformation model of soft tissues for real time surgical simulation. Information: An International Interdisciplinary Journal, 13(6):20112020, 11 2010.