Energy-Efficient Fault-Tolerant Scheduling Approach for Embedded Real Time Systems

Chafik Arar, Hamoudi Kalla, Salim Kalla, and Sonia Sabrina Bendib
Department of Computer Science, University of Banta, Algeria
Email: {Chafik.arar, hamoudi.kalla}@gmail.com, {salim.kalla,Bendib.SS}@univ-batna.dz

***Abstract*—In this paper, we propose a fault-tolerant scheduling heuristic that achieves low energy consumption and high reliability efficiency. Our scheduling algorithm is dedicated to multi-bus heterogeneous architectures, which take as input a given system description and a given fault hypothesis. It is based on active redundancy to mask a fixed number k of failures supported in the system, so that there is no need for detecting and handling such failures. In order to maximize the system's reliability, the replicas of each operation are scheduled on different reliable processors. Finally, we show with an example that our approach can maximize reliability and reduce energy consumption when using active redundancy.**

***Index Terms*—embedded systems, real time systems, scheduling, energy consumption, reliability, active redundancy, multi-bus**

I. Introduction

Today, embedded real-time systems invade many sectors of human activity, such as transportation and management, energy production, robotics, and telecommunication. The progresses achieved in electronics and data processing improves the performances of these systems. As a result, the new systems are increasingly small and fast, but also more complex and critical, and thus more sensitive to *faults*. The presence of some faults in these systems, accidental (design, interaction,…) as well as intentional (human, virus, …), are inevitable. Due to catastrophic consequences (human, ecological, and/or financial disasters) that could involve a fault, these systems must be fault-tolerant [1]–[3]. This is why fault-tolerant techniques are necessary to make sure that the system continues to deliver a correct service in spite of faults [4]–[9].

A fault can affect either the hardware or the software of the system. Thanks to formal validation techniques (e.g., model-checking, theorem proving, abstract interpretation, test case generation,…) a lot of software faults can be prevented. Although software faults are still an important issue, we chose to concentrate on hardware faults. More particularly, we consider only processors faults in systems with distributed architectures. Furthermore, as we are targeting embedded systems with limited resources (for reasons of weight, encumbrance, energy consumption, or price constraints), we investigate only software solutions. Several hardware fault-tolerant approaches [10]–[12] have been proposed in the literature. Because there are several methods of connecting processors (e.g., point-to-point, multipoint connections …), each of these approaches target a specific type of architecture. For instance [11] targets distributed architectures with point-to-point connections, while [12] targets distributed architectures with CAN (Controller Area Network) buses. A bus is a multipoint connection characterized by a physical medium that connects all the processors of the architecture. Buses, with broadcast or non-broadcast communications, are widely used in several areas, such as automotive or robotics. In the broadcast case, that we address in this document, each communication is received by all the processors, while in the non-broadcast case, only the destination processor receives the communication.

This document addresses hardware fault-tolerant approaches based on scheduling algorithms, to tolerate processors faults in distributed architectures. The approach that we propose is our most recent work for building fault-tolerant distributed embedded real-time systems. Prior results have been published in [13]–[17]. In this paper, we are interested in approaches based on scheduling algorithms that maximize reliability and reduce energy consumption when using active redundancy to tolerate processors.

The paper is organized as follows. Section II describes the system model and states the faults assumptions. Section III presents our approach for providing fault-tolerance. Section IV shows with an example how our approach can maximize reliability and reduce energy consumption when using active redundancy. Finally, Section V concludes the paper and proposes future research directions.

II. Models

A. Algorithm Models

The algorithm is modeled as a data-flow graph, called algorithm graph and noted ALG. Each vertex of ALG is an operation (task) and each edge is a data-dependence. A data-dependence, noted by →, corresponds to a data transfer between a producer operation and a consumer operation. o1→o2 means that o1 is a predecessor of o2,

Manuscript received July 1, 2012; revised September 10, 2013.

and o2 is a successor of o1. Operations with no predecessor (resp. no successor) are the input interfaces (resp. output), handling the events produced by the sensors (resp. actuators). Fig. 1 presents an example of an algorithm graph, with seven operations v1, v2, v3, v4, v5, v6 and v7.

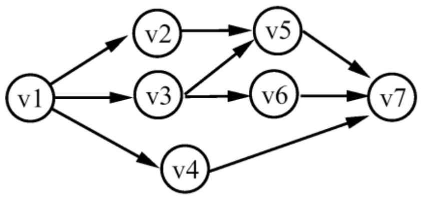

Figure 1. Algorithm graph.

B. Architecture Model

The architecture is modeled by a non-directed graph, noted ARC, where each node is a processor, and each edge is a bus. Classically, a processor is made of one computation unit, one local memory, and one or more communication units, each connected to one communication link. Communication units execute data transfers. We assume that the architecture is heterogeneous and fully connected. Fig. 2 is an example of ARC, with four processors P1, P2, P3 and P4, and three buses B1, B2 and B3.

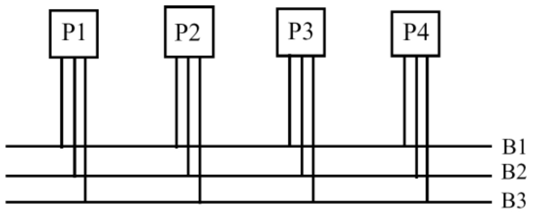

Figure 2. Architecture graph.

C. Distribution, Execution Time and Real-Time Constraints

The distribution constraints are the hardware preferences. They define the exclusion relationship between some hardware and some software components. As the architecture is heterogeneous, the execution time of each operation can vary from one processor to another. The temporal constraints define the worst execution times and the worst communication times respectively of the tasks and data communication on the architectures components (processors and communication bus).

Our real-time system is based on cyclic executive; this means that a fixed schedule of the operations of ALG is executed cyclically on ARC at a fixed rate. This schedule must satisfy one real-time constraint which is the length of the schedule. The temporal constraints define the worst execution times and the worst communication times respectively of the tasks and data communication on the architectures components (processors and communication bus). We associate to each operation oi a worst case execution time (WCET) on each processor pj of ARC, noted Exe(oi; pj). Also, we associate to each data dependency dpdi a worst case transmission time (WCTT) on each bus bj of the architecture, noted Exe(dpdi; bj).

D. Fault Model

We assume only hardware components failures and we assume that the algorithm is correct W.R.T. its specification, i.e., it has been formally validated, for instance with model checking and/or theorem proving tools. We consider only transient processors faults. Transient faults, which persist for a short duration, are significantly more frequent than other faults in systems [18]. Permanent faults are a particular case of transient faults. We assume that at most k processors faults can arise in the system, and that the architecture includes more than k processors.

III. The Proposed Approach

In this section, we first discuss the basic principles used in our solution, based on scheduling algorithms. Then, we describe in details our scheduling algorithm.

A. Principles

Active redundancy: in order to tolerate upto k arbitrary processors faults, our solution is based on active redundancy approach. The advantage of the active redundancy of operations is that the obtained schedule is static; in particular, there is no need for complex on-line re-scheduling of the operations that were executed on a processor when the latter fails; also, it can be proved that the schedule meets a required real-time constraint, both in the absence and in the presence of faults. In many embedded systems, this is mandatory. To tolerate upto k processors faults, each operation o of ALG is actively replicated on k+1 processors of ARC (see Fig. 3 and Fig. 4). We assume that all values returned by the k+1 replicas of any operation o of ALG are identical.

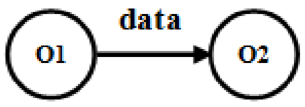

Figure 3. ALG with two operations.

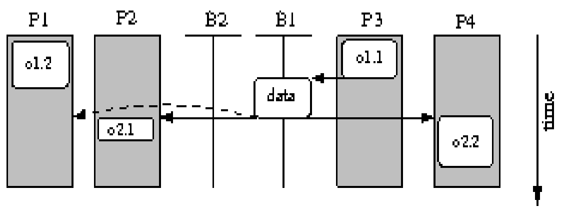

Figure 4. Active redundancy.

Voltage, frequency, and energy consumption: the maximum supply voltage is noted Vmax and the corresponding highest operating frequency is noted fmax. For each operation, its WCET assumes that the processor operates at fmax and Vmax (and similarly for the WCCT of the data-dependencies). Because the circuit delay is almost linearly related to 1/V, there is a linear relationship between the supply voltage V and the operating frequency f. In the sequel, we will assume that the operating frequencies are normalized, that is, fmax=1 and any other frequency f is in the interval (0,1). Accordingly, the execution time of the operation or data-dependency X placed onto the hardware component C (be it a processor or a communication link) running at frequency f (taken as a scaling factor) is:

$$\mathrm{Exe}(X, C, f) = \mathrm{Exe}(X, C) / f \qquad (1)$$

Concerning the power consumption, we follow the model of Zhu *et al*. [19]. For a single operation placed onto a single processor, the power consumption P is:

$$P = P_s + h(P_{ind} + P_d) \qquad (2)$$

$$P_d = C_{ef} V^2 f \qquad (3)$$

where P_s is the static power (power to maintain basic circuits and to keep the clock running), h is equal to 1 when the circuit is active and 0 when it is inactive, P_{ind} is the frequency independent active power (the power portion that is independent of the voltage and the frequency; it becomes 0 when the system is put to sleep, but the cost of doing so is very expensive), P_d is the frequency dependent active power (the processor dynamic power and any power that depends on the voltage or the frequency), C_{ef} is the switch capacitance, V is the supply voltage, and f is the operating frequency. For processors, this model is widely accepted for average size applications, where C_{ef} can be assumed to be constant for the whole application.

For a multiprocessor schedule S, we cannot apply directly equation (3). Instead, we must compute the total energy E(S) consumed by S, and then divide by the schedule length L(S):

$$P(S) = E(S) / L(S) \qquad (4)$$

We compute E(S) by summing the contribution of each processor, depending on the voltage and frequency of each operation placed onto it. On the processor pi, the energy consumed by each operation is the product of the active power $P^i_{ind} + P^i_d$ by its execution time.

In our approach, as k+1 replicas of each operation are scheduled actively on k+1 distinct processors, the energy consumed by the system is maximal. In order to reduce energy consumption, we propose to execute the k+1 replicas of an operation with different frequencies f. As all the k+1 replicas of an operation may have different end execution time (see Fig. 4 for the replicas o2.1 and o2.2), we choose to align the execution time of all the replica by changing the frequency f of each replica (see Fig. 5).

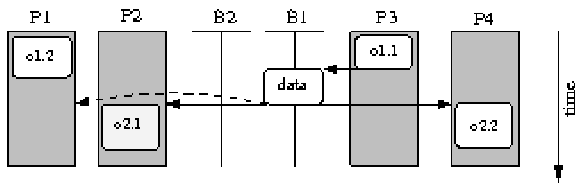

Figure 5. Changing the frequency of o2.1

Reliability: in order to compute reliability R, we propose to use the Global System Failure Rate per time unit (GSFR) function that we have proposed in [15]. The GSFR is the failure rate per time unit of the obtained multiprocessor schedule. Using the GSFR is very satisfactory in the area of periodically executed schedules. This is the case in most real-time embedded systems, which are periodically sampled systems. In such cases, applying brutally the exponential reliability model yields very low reliabilities due to very long execution times (the same remark applies also to very long schedules). Hence, one has to compute beforehand the desired reliability of a single iteration from the global reliability of the system during its full mission; but this computation depends on the total duration of the mission (which is known) and on the duration of one single iteration (which may not be known because it depends on the length of the schedule under construction). In contrast, the GSFR remains constant during the whole system's mission: the GSFR during a single iteration is by construction identical to the GSFR during the whole mission.

Our fault tolerance heuristic is GSFR-based to control precisely the scheduling of each replica of an operation from the beginning to the end of the schedule.

The GSFR of scheduling an operation oi, noted $\Lambda(S_n)$, is computed by the following equation:

$$\Lambda(S_n) = -\log R(S_n) / U(S_n) \qquad (5)$$

where S_n is the static schedule at step n of the algorithm, and $U(S_n)$ is the total utilization of the processors.

B. The Scheduling Algorithm

The principles of our approach are implemented by a scheduling algorithm, called Energy Fault Tolerant Heuristic (EFTH). It is a greedy list scheduling heuristic, which schedules one operation at each step (n). It generates a distributed static schedule of a given algorithm ALG onto a given architecture ARC, which minimizes the system's run-time, and tolerates upto k processors faults, with respect to the real-time and the distribution constraints. At each step of the greedy list scheduling heuristic, the pressure schedule function [15] is used as a cost function to select the best operation to be scheduled.

The algorithm of EFTH is divided into four steps: initialization step, selection step, a distribution and scheduling step, and finally an update step. The superscript number in parentheses refers to the steps of the heuristic, e.g., O^n_{sched}. The EFTH scheduling algorithm is described below:

```
Algorithm EFTH:
input: ALG, ARC, number of processors failures (k);
output: a fault-tolerant schedule;
Initialize the lists of candidate and scheduled operations:
    n := 0;
    $O^{(0)}_{cand}$ := { o ∈ O | pred(o) = \emptyset};
    $O^{(0)}_{sched}$ := ϕ ;
while $O^{(n)}_{cand} \neq \phi$ do
```

- For each candidate operation o_{cand}, compute its schedule pressure $\sigma^{(n)}$ and GSFR on each processor p_k.

$$\sigma^{(n)}(o_i,p_j) = S^{(n)}_{oi,pj} + S^{(n)}_{oi} - R^{n-1}$$

$$\Lambda(S_n) = -\log R(S_n) / U(S_n)$$

- For each candidate operation o_{cand}, select k+1 best processors P_{best} which minimizes $\sigma^{(n)}$ and GSFR.
- Select the most urgent candidate operation o_{urgent} between all o^{i}_{cand} of $O^{(n)}_{cand}$.
- Schedule the k replicas of o_{urgent} on P_{best};
- Align all the replicas of o_{urgent} by changing frequencies;
- Update the lists of candidate and scheduled operations:

$O^{(n)}_{sched} := O^{(n-1)}_{sched} \cup \{ o_{urgent} \}$;
$O^{(n+1)}_{cand} := O^{(n)}_{cand} - \{ o_{urgent} \} \cup \{o' \in succ(o_{urgent}) \mid pred(o') \subseteq O^{(n)}_{sched} \}$;

- n := n + 1;

end while
end

The set of candidate operations O_{cand} is initialized as the operations without predecessor. The set of scheduled operations O_{sched} is initially empty. In the selection step, a k+1 processors are selected among all the processor of ARC to schedule each replica of an operation. The selection rule is based schedule pressure and GSFR. The scheduled operation o_{best} is removed from O_{cand}, and the operations of ALG which have all their predecessors in the new set of scheduled operations are added to this set. In the schedule pressure function, $S^{(n)}_{oi,\ pj}$ compute the earliest time at which operation oi can start its execution on processor pj. The $R^{(n-1)}$ is the critical path length at step (n-1) (that is, before scheduling oi).

IV. Example

We have applied the EFTH heuristic to an example of an algorithm graph and an architecture graph composed of four processors and four buses. The algorithm graph is presented in Fig. 6. The failure rates of the processors are respectively 10^{-5}, 10^{-5}, 10^{-6} and 10^{-6}.

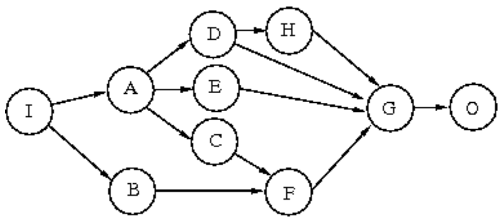

Figure 6. Algorithm graph.

Fig. 7 shows the non-fault-tolerant schedule produced for our example with a basic scheduling heuristic. (for instance the one of SynDEx). SynDEx is a tool for optimizing the implementation of real-time embedded applications on multicomponent architecture.

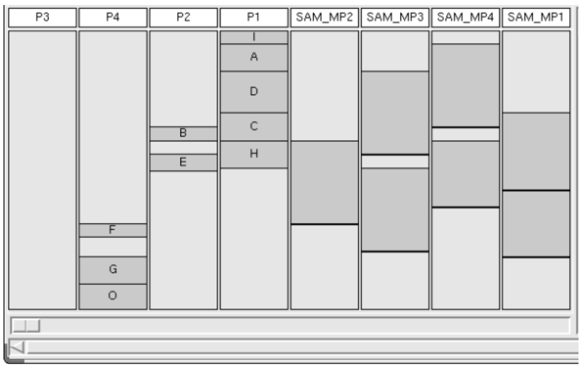

Figure 7. Schedule generated by SynDEx.

Fig. 8 shows the fault-tolerant schedule produced for our example with a EFTH scheduling heuristic without changing frequencies. The schedule length generated by this heuristic is 18.6. The GSFR of the non-reliable schedule is equal to 0.0000264. The energy E is equal to 32.3.

Fig. 9 shows the fault-tolerant schedule produced for our example with a EFTH scheduling heuristic. The schedule length generated by this heuristic is 25.9. The GSFR of the non-reliable schedule is equal to 0.0000251. The energy E is equal to 24.75.

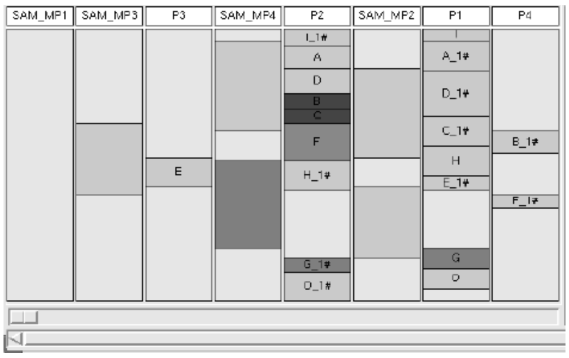

Figure 8. EFTH without changing frequencies

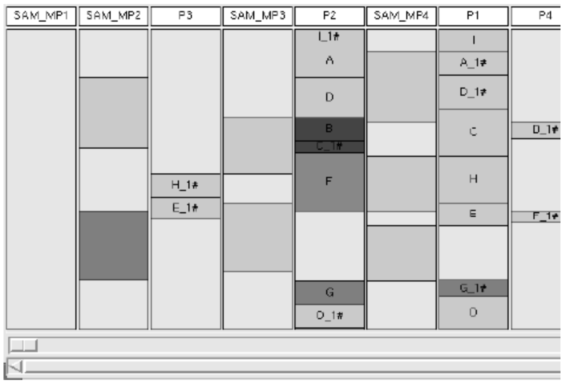

Figure 9. A schedule generated by EFTH

V. Conclusion

We have proposed in this paper a solution to tolerate several processors faults in distributed heterogeneous architectures with multiple-bus topology. The proposed solution, based on active redundancy, is a list scheduling heuristic called EFTH. It generates automatically a distributed static schedule of a given algorithm onto a given architecture, which minimizes the system's run-time, and tolerates upto k processors faults, with respect to real-time and distribution constraints. The scheduling strategy based on variable frequency minimizes energy consumption. Currently, we are working on a new solution to take into account communication failures into account.

REFERENCES

[1] A. Avizienis, "Dependable systems of the future: What is still needed?" in *Proc. IFIP World Computer Congress*, 2004, pp. 79-90.

[2] A. Avizienis, J. C. Laprie, and B. Randell, "Dependability and its threats: A taxonomy," in *Proc. IFIP World Computer Congress*, 2004, pp. 91-120.

[3] N. Suri and K. Ramamritham, "Editorial: Special section on dependable real-time systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 10, no. 6, pp. 529-531, June 1999.

[4] A. Avizienis, "Design of fault-tolerant computers," in *Proc. Fall Joint Computer Conference*, November 1967, pp. 733-743.

[5] A. Burns and A. Wellings, *Real-Time Systems and Programming Languages*, Addison-Wesley, 1997.

[6] P. Jalote, *Fault-Tolerance in Distributed Systems*, Prentice Hall, New Jersey, 1994.

[7] J. C. C. Laprie, A. Avizienis, and H. Kopetz, *Dependability: Basic Concepts and Terminology*, New York: Springer-Verlag, 1992.

[8] N. Suri and K. Ramamritham, "Editorial: Special section on dependable real-time systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 10, no. 6, pp. 529–531, June 1999.

[9] W. Torres-Pomales, "Software fault-tolerance: A tutorial," *National Aeronautics and Space Administration, Technical Report*, October 2000.

[10] K. P. Gummadi, M. J. Pradeep, and C. S. Ram Murthy, "An efficient primary-segmented backup scheme for dependable real-time communication in multihop networks," *IEEE/ACM Trans. on Networking*, vol. 11. no. 1, pp. 81-94, February 2003.

[11] X. Qin, H. Jiang, and D. R. Swanson, "An efficient fault-tolerant scheduling algorithm for real-time tasks with precedence constraints in heterogeneous systems," in *Proc. International Conference on Parallel Processing*, Canada, August 2002, pp. 360-386.

[12] J. Rufino, "Dual-media redundancy mechanisms for can," *Technical Report, Center for Computer Systems and Telematics*, Portugal, January 1997.

[13] I. Assayad, A. Girault, and H. Kalla, "Tradeoff exploration between reliability, power consumption, and execution time for embedded systems, the TSH tricriteria scheduling heuristic," *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 3, pp. 229-245, June 2013.

[14] I. Assayad, A. Girault, and H. Kalla, "Tradeoff exploration between reliability, power consumption, and execution time," in *Proc. International Conference on Computer Safety, Reliability and Security*, Naples, Italy, September 2011, pp. 437-451.

[15] A. Girault and H. Kalla, "A novel bicriteria scheduling heuristics providing a guaranteed global system failure rate," *IEEE/ACM Trans. on Dependable and Secure Computing*, vol. 6, no. 4, pp. 241-254, October 2009.

[16] M. Bachir and H. Kalla, "A fault tolerant scheduling heuristics for distributed real time embedded systems," presented at International Conference on Systems and Information Processing, Guelma, Algeria, May 2013.

[17] I. Assayad, A. Girault, and H. Kalla, "Scheduling of real-time and reliable embedded systems under reliability and power constraints," presented at International Conference on Complex Systems, Agadir, Morocco, November 2012.

[18] M. Pizza, L. Strigini, A. Bondavalli, and F. Di Giandomenico, "Optimal discrimination between transient and permanent faults," in *Proc. 3rd IEEE High Assurance System Engineering Symposium*, Bethesda, USA, 1998, pp. 214-223.

[19] D. Zhu, R. Melhem, D. Mosse, and E. Elnozahy, "Analysis of an energy efficient optimistic TMR scheme," in *Proc. International Conference on Parallel and Distributed Systems*, Newport Beach, USA, July 2004, pp. 559-568.

Chafik Arar received the Master degree in Computer Science from the University of Batna (Algeria) in 2005. He is currently a PhD candidate with Computer Science Department in the same university. His research topic is reliability and communication fault tolerance for real-time distributed embedded systems.

Hamoudi Kalla received the PhD degree from the National Polytechnic Institute of Grenoble in 2004. From 2005 to 2006, he was a postdoctoral fellow at the French National Institute for Research in Computer Science and Control (INRIA) in the ESPRESSO Project Team, Rennes. He is currently an assistant professor in the REDS Team, Department of Computer Science, University of Batna, Algeria. His current research area focuses on developing reliability and fault tolerance techniques for distributed real-time embedded systems and on the formal verification of embedded-system-based intellectual property (IP) components.

Salim Kalla received the Master degree in Computer Science from Orleans University (France) in 2002. He is currently an assistant professor and a PhD candidate with Computer Science Department, University of Batna, Algeria. His research topic is scheduling and fault tolerance for critical real-time and Distributed embedded systems.

Sonia Sabrina Bendib received the Master degree in Computer Science from the University of Batna. She is currently an assistant professor and a PhD candidate with Computer Science Department in the same university. Her special fields of interest include Bi-criteria Scheduling Algorithms, reliability and fault tolerance in real-time embedded systems.