Energy-Aware Algorithms for Tasks and Bandwidth Co-Allocation under Real-Time and Redundancy Constraints

Francesco Prosperi, Mario Bambagini, Giorgio Buttazzo, Mauro Marinoni, Gianluca Franchino
{name.surname}@sssup.it
Scuola Superiore Sant'Anna, Pisa, Italy

***Abstract*—The energy consumption in distributed systems depends on several inter-related factors, including task partitioning, process redundancy, fault tolerance, task and message scheduling, and communication bandwidth allocation. Although some of these issues have been considered in the literature in isolation, a systematic approach considering all the constraints is still missing. This paper addresses the problem of allocating a task set and the required communication bandwidth on a distributed embedded system, aiming at reducing energy consumption while guaranteeing timing and redundancy constraints. Two heuristic approaches are proposed and compared against a complete method and simulated annealing. Simulation results show the effectiveness of the proposed approaches.**

I. Introduction

Today, a large number of embedded systems consists of a set of computing nodes interconnected via wired or wireless networks. In such systems, both node responsiveness and energy consumption are affected by the policies used to allocate and schedule tasks on the various nodes, schedule messages over the network, and manage the other available resources, such as the I/O devices.

In several situations (e.g., wireless sensor networks) redundancy is required to guarantee high quality and reliability in events detection. In fact, sensor nodes are typically affected by high failure rates, so that a certain level of replication is required in the system to increase reliability in the measurements. Another context where redundancy issues are relevant is an assembly line consisting of many robots in charge of performing and supervising the various production stages, where each robot may include different sensors, actuators and control units that need to exchange data and signals. Since a fault may jeopardize the quality of products, sensor redundancy would increase reliability and allow earlier failure detection and avoidance. Another benefit that comes from redundancy is the possibility of increasing measurement accuracy, where multiple observations from different nodes are integrated to produce more reliable aggregated values, used to reduce statistical uncertainty in the measurements.

While redundancy improves reliability and precision, it increases energy consumption due to CPU utilization, network communication, etc. The problem is complicated by the fact that there are situations in which reducing the energy consumption on a node may increase the energy dissipated in other nodes, so shortening the lifetime of the entire system. A way to address this issue is to consider the problem of co-scheduling tasks and messages with the objective of minimizing energy consumption while meeting timing constraints.

Contributions: This paper addresses the problem of partitioning tasks and communication messages on a distributed embedded system with the objective of guaranteeing timing constraints while minimizing the overall energy consumption and maximizing task duplication.

A trade-off between such two conflicting objectives is explored, ensuring task real-time requirements and guaranteeing a minimum number of task copies and system lifetime. Bandwidth and communication buffer constraints are also taken into account. Two heuristic approaches are presented and evaluated with respect to simulated annealing and a complete method based on branch and bound search.

Organization of the paper: Section II presents the system model, in terms of processing nodes, communication bandwidth, power model, computational and communication workload. The problem statement is introduced in Section III, including the specification of all the system constraints and the performance metric used to evaluate the goodness of an allocation. Section IV illustrates the proposed approaches, whereas Section V reports the experimental results. Finally, Section VI ends the paper with the concluding remarks.

A. Related work

Although a lot of research has been focused on scheduling on networked/multicore embedded systems [5], little work has been done to optimize allocation taking multiple objectives into account, such as real-time constraints for tasks and messages, energy consumption, and redundancy issues.

Kim et al. [11] considered the problem of mapping multimedia applications onto a multi-processor system-on-chip (MPSoC) and proposed a framework for design space exploration based on heterogeneous scheduling policies and DPM techniques. Schranzhofer et al. [19] addressed a similar problem proposing a dynamic mapping strategy able to minimize energy consumption to prolong the system lifetime for heterogeneous processing units.

Aydin et al. [2] proposed several methods for reducing the overall energy consumption exploiting frequency scaling in partitioning-based systems where EDF is the scheduler used for each node. More precisely, they modified and compared several well-known heuristics to consider the energy consumption into the performance metric.

Kandhalu et al. [8] proposed an approach for reducing the energy dissipation in multi-core platforms with a single voltage/frequency domain using a partitioned fixed-priority scheduler. They showed that balancing the workload is crucial as the core with the highest load is likely to impose the common frequency.

Other partitioned approaches take into account precedence dependencies among tasks [20], [15]. The major difficulty in this problem is to take precedence relations into account in the schedulability analysis. Buttazzo et al. [4] approached this issue by assigning intermediate deadlines to tasks according to the dependency graph, and then scheduling them according to EDF. In this context, other authors [9], [18] applied genetic algorithms to allocate tasks and generate a static schedule using DVFS techniques to reduce energy consumption. Huang et al. [7] addressed the problem of allocating and scheduling dependent tasks on a set of heterogeneous cores also considering communication costs. Three solutions were compared to solve the problem, one based on Integer Linear Programming (ILP) and two based on simulated annealing.

Concerning task redundancy, timing and precedence constraints, Ramamritham [17] proposed an approach that guarantees the allocation of a given number of subtask copies, where each task is described by a precedence graph.

Auluck and Agrawal [1] proposed a method for allocating and scheduling non-preemptive tasks on a heterogeneous system to guarantee the execution of a real-time task set in case of a processor fault. Once the task graph has been allocated, the algorithm keeps allocating selected task duplicates until all the processors are utilized.

Qin et al. [16] proposed a fault-tolerant algorithm to handle a single processor failure in a fully connected network of heterogeneous nodes running real-time dependent tasks. In their model, failure conditions are supposed to be detected after a fixed period.

Other researches focused on networked embedded systems. Xie et al. [21] considered a single-hop network composed by heterogeneous devices without DVS capability and provided a task allocation method aimed at reducing the latency and the energy consumption of the system, taking into account both computation and communication activities.

Kim et al. [10] proposed a method to dynamically assign tasks in a single-hop wireless network of mobile nodes with DVS mechanism, guaranteeing time constraints and a minimum lifetime of eight hours. To this end, the authors proposed and compared a set of dynamic mapping heuristics.

Xue et al. [22] considered a network-wide dynamic energy management mechanism for a system composed by wireless nodes with DVS capability, executing real-time tasks with precedence constraints. The proposed method assigns tasks and DVS levels to each node, taking into account both radio sleep intervals and the application parallelism.

Grid computing systems with mobile nodes have been considered by Chunlin and Layuan [13], who proposed a method to guarantee both lifetime and time constraints of grid applications. The authors modeled the problem as a distributed energy constrained resources allocation, and proposed a price-based algorithm to provide an optimal solution.

None of the papers existing in the literature, however, addressed all the constraints considered in this work, hence they cannot be compared with the proposed approaches.

II. System Model

This work considers a distributed system of *m* homogeneous nodes $\Psi_1, \ldots, \Psi_m$ logically connected as illustrated in Figure 1. Each node performs sensory acquisition and sends messages to a *coordinator* node (Ψ_0), according to a star topology, using a TDMA-based communication protocol that guarantees a bounded transmission delay. In each TDMA time wheel, each node Ψ_i is assigned a bandwidth slot of length w_i. Note that, the *coordinator* Ψ_0 is not subject to the analysis carried out in this work, since it is assumed to be always on and available to the nodes.

Since the system is intended as a network of nodes, throughout the rest of the paper, the terms *system* and *network* are used interchangeably.

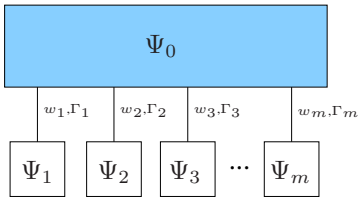

Figure 1. A sample network with a node coordinator.

The system is assumed to run an application Γ consisting of a set of *n* sporadic real-time tasks $\Gamma = \{\tau_1, \ldots, \tau_n\}$. Each task τ_j is characterized by a worst-case execution time C_j, a minimum inter-arrival time T_j, and a relative deadline D_j (assumed to be equal to T_j). Each task generates an infinite sequence of jobs, $\tau_{j,k}$, each having release time $r_{j,k}$ and absolute deadline $d_{j,k} = r_{j,k} + D_j$. Each job $\tau_{j,k}$ sends a message of payload M_j to node Ψ_0 at its termination, which can occur any time in the interval $(r_{j,k}, r_{j,k} + D_j]$. The overhead of transferring a message generated by a task into the transmission buffer is accounted in C_j. The utilization of task τ_j is denoted by u_j and is computed as C_j/T_j.

Each node Ψ_i hosts a subset Γ_i of Γ. The utilization of node Ψ_i and the system utilization are denoted as U_i and

U_{tot}, respectively. Note that a task τ_j can run in more than one node, but each node can execute at most one copy of τ_j. The number of running instances of task τ_j on the entire network is denoted as μ_j.

For each task τ_j, the application specifies a *reward function* denoted as γ_j, which grows with the number of task instances and, hence, measures the satisfaction of the task redundancy across the system.

Tasks in each node are scheduled by the Earliest Deadline First (EDF) [14] algorithm, but the analysis can easily be extended to different scheduling policies.

The timing parameters introduced above are summarized in Figure 2.

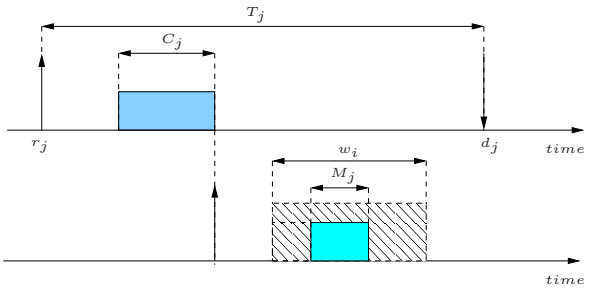

Figure 2. Timing parameters.

A. Bandwidth model

The communication between the nodes and the *coordinator* is managed by a TDMA-based communication protocol, according to a star topology. Each node Ψ_i is assigned a time slot w_i in each TDMA wheel where it can transmit its messages. The messages produced by the tasks running in node Ψ_i are enqueued in a transmission buffer of length Θ_i adopting a FIFO policy and then sent as soon as the bandwidth becomes available.

A necessary condition to guarantee that each node can transmit its messages without overflowing the transmission buffer requires the wheel length W to be no shorter than the minimum period in the task set, hence W is set as

$$W = \min_{j \in [1..n]} T_j. \tag{1}$$

Each task τ_j of node Ψ_i is assigned a time budget Q_j sufficient to ensure the correct delivery of the related message within $2T_j$ from the associated job release time $r_{j,k}$. Then, w_i is set as the sum of the budgets Q_j's of the tasks allocated to node Ψ_i:

$$w_i = \sum_{\tau_j \in \Gamma_i} Q_j. \tag{2}$$

The task budget Q_j is computed as follows:

$$Q_j = \frac{M_j}{\left\lfloor \frac{T_j}{W} \right\rfloor}. \tag{3}$$

Observe that $\left\lfloor \frac{T_j}{W} \right\rfloor$ represents the number of complete TDMA wheels available during a task period T_j, i.e., the number of time slots that the node running τ_j can exploit to transmit the message generated by such a task. Since M_j is the length of any message generated by τ_j, the rationale behind the computation of Q_j is to assign a time budget sufficient to send a message of length M_j every T_j time units.

If w^{com} denotes the sum of the slots w_i assigned to the nodes, the overall communication load on the network is w^{com}/W. Since for each node Ψ_i, the time slot w_i is computed in such a way the node can send all messages generated by its tasks, it follows that a necessary and sufficient condition to guarantee all message deadlines is:

$$w^{com} \leq W. \tag{4}$$

To save energy, the adopted bandwidth scheme reserves a slot $S = W - w^{com}$ over each W in which the communication is not allowed and all the nodes turn their transceiver off.

Consider a sample scenario featuring three nodes (Ψ_1,Ψ_2,Ψ_3) and a task set composed by two tasks (τ_1, τ_2), where node Ψ_1 hosts both the tasks, node Ψ_2 hosts τ_1, and node Ψ_3 hosts τ_2. The resulting TDMA wheel partition is illustrated in Figure 3. Note that, throughout the paper, the slot lengths w_i are ordered by the node indexes inside the wheel, while the budgets Q_j inside the slot are ordered by the task indexes.

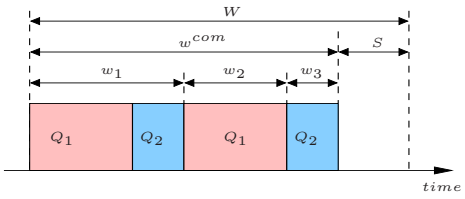

Figure 3. Bandwidth allocation example.

B. Power Model

Since this paper focuses on a network-wide approach for reducing energy consumption, a simplified power model is adopted for the nodes to estimate their power consumption. A finer model would be unnecessarily detailed in this context, considering that each node could apply further runtime strategies to save energy.

In particular, each device (i.e., CPU or transceiver) is assumed to be in one of the following states:

- *active*. In this state, the device performs its job executing tasks or handling messages. The power consumed in this state is denoted by P_a^{CPU} and P_a^{com}, respectively.
- *sleep*. In this state, the device is completely turned off and consumes the least amount of power, denoted as P_s^{CPU} and P_s^{com}, respectively.

The overhead to switch between power states for both the devices is assumed to be negligible, both in terms of time and energy.

The node power consumption is estimated as follows:

$$P_i = U_i P_a^{CPU} + (1-U_i) P_s^{CPU} + \frac{w_i}{W} P_a^{com} + \left(1 - \frac{w_i}{W}\right) P_s^{com}. \quad (5)$$

The node and system mean power consumption is estimated assuming that each node exploits the idle intervals in task and message schedules turning the CPU and the transceiver off during idle periods and outside bandwidth slots, respectively, as follows:

$$P = U_{tot} P_a^{CPU} + (m - U_{tot}) P_s^{CPU} + \frac{w^{com}}{W} P_a^{com} + \left(m - \frac{w^{com}}{W}\right) P_s^{com}. \quad (6)$$

The current energy level of node Ψ_i is denoted by E_i and every node is provided with the same initial energy $E^{(0)}$.

C. Lifetime model

The system lifetime is a key element for evaluating sensor networks. Its definition is not univocal because it depends from a lot of application parameters (e.g., number of sensors, network coverage, quality of service, etc) and its value is an aggregation of all nodes statuses. The network can only fulfill its purpose as long as it is considered *alive*, but not after that. Therefore, the lifetime is an indicator for the maximum utility a sensor network can provide.

There are a lot of lifetime definitions [6], but the most common and most frequently used in the literature is the m-of-m lifetime. In this definition, the distributed system lifetime L ends as soon as the first node fails, thus $L = \min_i L_i$, with L_i representing the lifetime of node Ψ_i computed as E_i/P_i.

A common constraint on the lifetime imposes a minimum amount of time ($\overline{L}$) in which the system has to stay alive. The minimum lifetime could be formalized as:

$$\min_i \frac{E_i}{P_i} > \overline{L}.$$

D. Allocation specific parameters

An *allocation* describes the distribution of the tasks instances among the nodes of the system. Since a task τ_j can run on different nodes simultaneously, an allocation matrix $A \in \{0,1\}^{m \times n}$ is defined to keep track of such a distribution on the network. The generic element a_{ij} of the allocation matrix A is a boolean variable indicating whether task τ_j is present on node Ψ_i or not. Note that each node can execute at most one copy of τ_j. In the following, a set of previously defined parameters are formalized based on A.

The subset Γ_i of tasks running on node Ψ_i is expressed as

$$\Gamma_i = \{\tau_j | a_{ij} = 1\}, \quad (7)$$

while the length w_i of the communication bandwidth slot assigned to the node Ψ_i is computed as

$$w_i = \sum_{j=1}^{n} a_{ij} Q_j. \quad (8)$$

The total utilization U_i of node Ψ_i and the system utilization U_{tot} are formally defined as

$$U_i = \sum_{j=1}^{n} a_{ij} u_j \quad (9)$$

and

$$U_{tot} = \sum_{i=1}^{m} U_i. \quad (10)$$

The actual number μ_j of instances of task τ_j across the system is computed as

$$\mu_j = \sum_{i=1}^{m} a_{ij}. \quad (11)$$

The worst case scenario for the transceiver buffer occurs when two instances of the same task generate a message between the end of the previous Q_j and the start of the next one, as depicted in Figure 4. In the example illustrated in the figure, the Q_j is supposed to occur at the beginning of each wheel. More than two generations are not possible because the wheel is equal to the minimum task period. This leads to consider that the minimum required size Θ_i^{min} of the transceiver internal buffer on node Ψ_i is computed as:

$$\Theta_i^{min} = 2 \sum_{j=1}^{m} a_{ij} M_j.$$

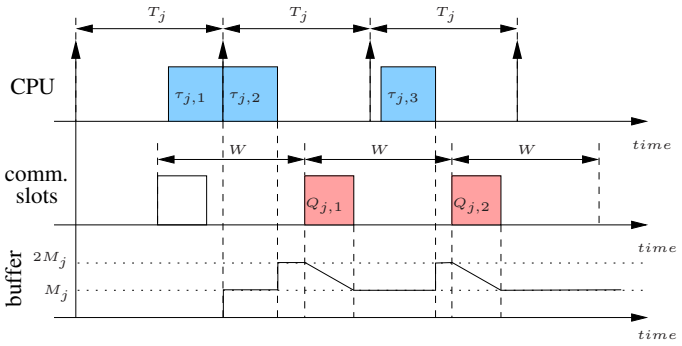

Figure 4. Example of worst-case buffer usage.

III. Problem Statement

This paper addresses the problem of allocating a real-time task set over a distributed system composed by homogeneous embedded nodes. Such an allocation has to meet a set of constraints:

Task schedulability: All the tasks running in each node have to terminate within their deadlines. Under the assumption that all the tasks have relative deadlines equal to periods and are scheduled by EDF, such a constraint is expressed as follows:

$$\forall i \in [1,m] \quad U_i \leq 1. \tag{12}$$

Bandwidth: To guarantee the schedulability of the messages produced by the tasks, the sum w^{com} of all the slots assigned to the nodes must not exceed the wheel length W:

$$w^{com} \leq W. \tag{13}$$

Buffer: For each node, the transmission buffer (whose length Θ is equal for each node) must be long enough to contain all the messages generated in the worst case, that is:

$$\forall i \in [1,m] \quad \Theta \geq \Theta_i^{min}. \tag{14}$$

Redundancy: To improve the accuracy level of measurements, a minimum number μ_j^{min} of running instances for each task τ_j must be guaranteed to be executed in the system, that is:

$$\forall j \in [1,n] \quad \mu_j \geq \mu_j^{min}. \tag{15}$$

Lifetime: To guarantee a desired system lifetime $\overline{L}$, the initial energy $E^{(0)}$ available in each node must be sufficient to keep the entire network alive for the required duration, that is:

$$\forall i \in [1,m] \quad \frac{E^{(0)}}{P_i} \geq \overline{L}. \tag{16}$$

A. Performance evaluation

To evaluate the performance of a generic allocation, three distinct normalized indexes ($\in [0,1]$) are introduced to measure redundancy, energy saving, and uniformity. Such factors are combined to obtain a normalized scalar function able to compare the goodness of different allocations.

The performance function is modeled to operate also in the case of unfeasible allocations. Such a function is built so the co-domain range of the performance function is $[-3,1]$, equally distributed between the feasible situations ($[0,1]$) and the unfeasible ones ($[-3,0)$).

As already explained in Section II, more instances of each task can execute on the network. Note that an implicit limit on the running instances of a task τ_j exists as at most *m* nodes can run a single instance of τ_j.

For each task τ_j, a monotonic function γ_j is provided by the application designer to specify the reward according to the actual number of task instances running in the system, from μ_j^{min} to m. γ_j is a function with bounded output value in $[0,1)$. The redundancy index ρ is defined as

$$\rho \triangleq \frac{\sum_{j=1}^{n} \gamma_j}{n}.$$

The overall energy consumption of the system is a key parameter taken into account by the analysis carried out in this work. The index ξ measures the energy saving of an allocation with respect to the highest power consumption P_{max} that the system can experience, that is:

$$\xi \triangleq \frac{\max(0, P_{max} - P)}{P_{max}}.$$

Notice that P_{max} refers to an ideal allocation in which all the nodes are fully loaded (i.e. all nodes have utilization equal to 1) and the bandwidth slots are assigned in such a way that w^{com} is equal to W.

Based on the definition of system lifetime assumed in this work, how task instances are spread across the system nodes is of key importance, as a more uniform load distribution results in a longer global lifetime (due to the proportionality between utilization and power consumption at node level).

The utilization uniformity of the allocation is evaluated through the variance σ^2 of the total utilization U_i of each node Ψ_i computed as follows:

$$\sigma^2 \triangleq \frac{1}{m} \sum_{i=1}^{m} \left(U_i - \frac{U_{tot}}{m} \right)^2.$$

As for the previous indexes, the uniformity of the allocation is evaluated through an index α normalized in the interval $[0,1]$:

$$\alpha \triangleq \frac{\max(0, \sigma_{max}^2 - \sigma^2)}{\sigma_{max}^2},$$

where σ_{max}^2 denotes the variance in the worst-case scenario in which half nodes are fully loaded and half are empty, leading to a value of σ^2 equal to 0.25.

Considering the trade-off between redundancy and power consumption, a parameter $\eta \in [0,1]$ is introduced to balance the two contributions. Notice that such a parameter remains constant throughout all the optimization process.

For a feasible allocation, the performance function Φ is defined as follows:

$$\Phi_\eta^+(A) = \alpha[\eta\rho + (1-\eta)\xi], \tag{17}$$

whereas, for an unfeasible allocation, it is defined as follows:

$$\begin{aligned} \Phi^-(A) \quad = \quad & -\frac{1}{n} \sum_{j}^{n} \frac{\max(0, \mu_j^{min} - \mu_j)}{\mu_j^{min}} + \\ & -\frac{\sum_i^m \max(0, U_i - 1)}{(U_{tot} - 1)m} + \\ & -\frac{\max(0, w^{com} - W)}{m \sum_j^n Q_j}. \end{aligned} \tag{18}$$

Such a function is composed by three terms, each introduced to weigh the violation of one of the evaluated constraints. Note that while the second term is defined at a node level, since it concerns CPU overrun, the last term operates at a system level, as the communication wheel is shared among all the nodes.

IV. Proposed Algorithms

This section presents three types of algorithms that try to maximize a user-defined performance function under the constraints formalized in Section III.

The cost of each approach is measured through the number of performance evaluations, rather than through the total processing time, since the execution time of an algorithm is affected by several factors (e.g., processor speed, memory, and implementation efficiency). The complexity to evaluate a specific allocation is $O(nm)$.

A. Complete Search

A branch and bound search has been implemented to evaluate the distance of the optimal solution with respect to the other methods, at least for a small system size. The exhaustive search uses an EDF feasibility test on single nodes to prune unfeasible branches.

Note that, at low total utilizations, the number of feasible (single node) allocations is exponential on the number of tasks, since very low tasks utilizations make any task combination feasible, and drops down at higher utilizations.

Since the embedded nodes composing the system are homogeneous, several solutions are symmetric, thus leading to the same performance. To avoid the generation of symmetric allocations, this approach combines the solutions previously found to reduce the number of possible configurations.

B. Heuristics

This section presents two heuristic approaches, *Heuristic A* and *Heuristic B*.

Heuristic A starts from an empty allocation and, at each step, generates all the possible configurations that differ from the current one by a single task (by switching a single element of matrix A from 0 to 1). Then, the configuration with the highest performance is selected. In the case of multiple configurations featuring the highest performance, the first occurrence is selected. The algorithm stops as soon as it fails to improve the current best performance.

Since at each step the approach performs at most $n \times m$ evaluations, its worst-case complexity is $O(n^3m^3)$, as it is possible to insert at most $n \times m$ instances to fill the A matrix.

Heuristic B is a variation of Heuristic A, since at each step it selects the node with the lowest utilization, testing the resulting performance at each addition. Since at each step the approach has to evaluate n different configurations, the worst-case complexity is $O(n^3m^2)$. Like Heuristic A, the approach stops as soon as it fails to improve the current best performance.

C. Simulated Annealing

Simulated Annealing [12] is an effective technique for finding an acceptably good solution in a fixed amount of time, even in large search spaces. In this approach, a generic point s of the search space is called a $state$, and the neighborhood of a state s is defined as the set of states produced from s by suitable alterations.

In this implementation, a state is represented by a global allocation, and its neighborhood is the set of allocations that differ from the starting one only for one boolean element of the matrix A. This means that the neighborhood of a state is another allocation in which a single task is added (if it was not present) or removed (if it was present). The other parameters, that is the cooling factor, the maximum number of tries at fixed temperature, and the temperature threshold, are selected to improve performance.

In this work, the simulated annealing has been implemented according to two approaches, which differ for their initial state. The first approach starts from the empty allocation matrix and it is referred to as *Basic Simulated Annealing* (BSA), while the second one lies upon the result of Heuristic A and, hence, it is referred to as *Heuristic Simulated Annealing* (HSA).

V. Experimental Results

This section presents a set of experimental results of the approaches proposed in Section IV. Such results are obtained by simulation on synthetic workloads adopting the power profile taken by the Microchip dsPIC[1] datasheet, interpolating the typical consumptions. To test the behavior of the various approaches, a C simulator has been developed.

An execution scenario is characterized by the tuple (n, m, U, B), where n denotes the number of tasks, m the number of nodes in the network, U the total task set utilization, and B the total communication bandwidth required by the task set ($B = \sum_{j=1}^{n} M_j/T_j$).

Given a total utilization factor (U and B), the task set features (C_j, T_j and M_j) are computed according to a uniform distribution [3]. From such values, the wheel length W and task bandwidth slots (Q_j) are computed according to Equation (1) and Equation (3).

As already exposed in Section III-A, the application designer has to specify the reward function γ_j for each task τ_j. If μ_j^{sat} denotes the number of running instances of τ_j beyond which the measures have no gain on accuracy, in this set of experiments γ_j is set as

$$\gamma_j = \left(1 - e^{\Delta_j}\right),$$

where

$$\Delta_j = \frac{-5(\mu_j - \mu_j^{min})}{\mu_j^{sat} - \mu_j^{min}}.$$

For all the experiments, μ_j^{sat} is set equal to the number of nodes m. The balancing parameter η between redundancy and energy saving is chosen to be 0.5.

Due to the complexity of considering all the parameters introduced in the analysis, the set of experiments described in this section do not check the violation of the constraints

[1] dsPIC33FJ256MC710 microcontroller

on the minimum buffer and the minimum lifetime formalized in Equation (14) and Equation (16), respectively.

The first experiment evaluates the complexity of the algorithms as a function of the product $n \times m$, for all the approaches, setting $U = m/2$ and $B = 0.3$. Table I reports the number of performance evaluations averaged on 50 runs of each approach. As described in Section IV-A, the Complete Search (CS) approach represents an effective method to find the optimal solution only for small problem size, as the number of evaluations becomes too high already at $n \times m$ = 10x6. The number of evaluations performed by each approach reflects the complexity values reported in Section IV. Simulated annealing techniques have a higher number of evaluations since they explore the search space until the system is cooled but with a conditional stop criterion on the stability of the result. The performance difference between CS and the other approaches is under 5%.

$n \times m$	CS	Heu A	Heu B	BSA	HSA
5x4	519	161	40	14684	20603
10x6	$4.8 \cdot 10^9$	1186	168	22561	50234
15x8	-	4036	432	43598	63678
20x8	-	7139	787	57663	79626
25x10	-	15101	1309	91628	126059

Table I
PROBLEM COMPLEXITY.

The second experiment evaluates the performance index Φ as a function of the task set utilization U, and results are shown in Figure 5. All the approaches have been tested with $n = 25$, $m = 10$ and $B = 0.3$. The utilization is normalized on the number of nodes m.

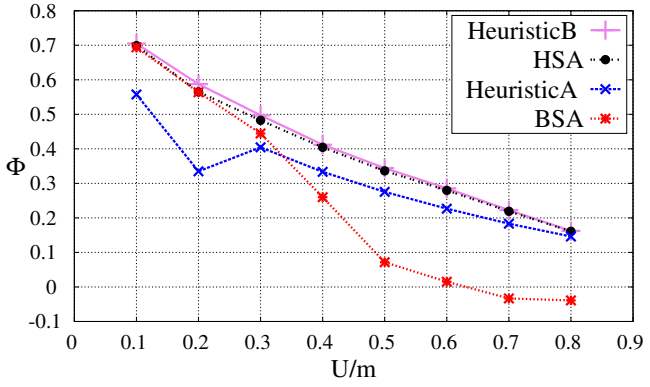

Figure 5. Performance analysis of the approaches.

As Figure 5 shows, Heuristic B outperforms all the approaches. BSA performs poorly at utilizations higher than 0.5. Heuristic A is often able to find a reasonable solution, although it can be improved by HSA, whose performance is very similar to that of Heuristic B.

The third experiment deeply investigates Heuristic B, analyzing the three components of the performance index as a function of the utilization U, normalized on the number of nodes m.

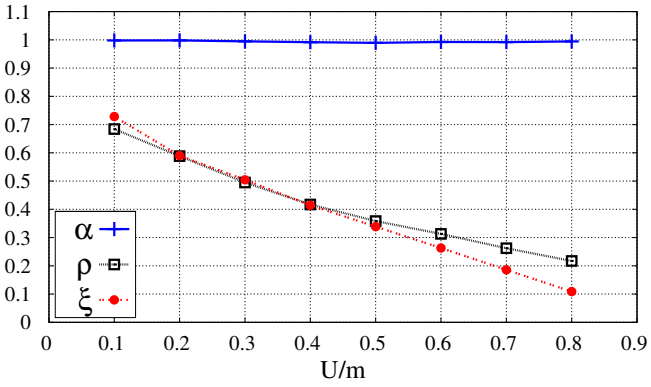

Figure 6. Performance components.

The results are reported in Figure 6, which clearly shows that the α factor is about constant, revealing that the solutions found by Heuristic B are characterized by uniform allocations. As expected, both the redundancy and energy saving indexes decrease with the growing of the normalized CPU load. This occurs because tasks with higher utilization are difficult to spread across the network without violating the schedulability constraint and causing a sensible increase of power consumption.

The fourth and last experiment analyzes Heuristic B as a function of both n and m, for a bandwidth utilization $B = 0.3$. The values of n range in $\{5, 10, 15, 20, 25\}$, while the values of m range in $\{4, 6, 8, 10, 12\}$. In this experiment, the utilization U is always set to $m/2$.

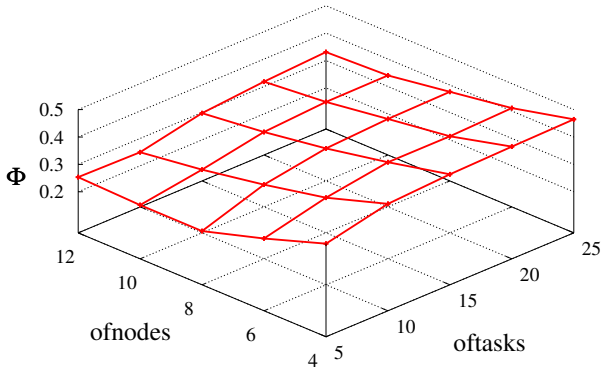

Figure 7. Performance analysis of Heuristic B when B=0.3.

As reported in Figure 7, the performance obtained by Heuristic B improves as the number of tasks increases. This is due to the higher fragmentation of the task set, that results in a wider feasible solution space. Instead, the performance index decreases as the number of nodes grows. Since the utilization U is always set to the half of m, the tasks result

in a higher utilization. As for the third experiment, the approach is able to spread only a few tasks instances across the network paying a higher power consumption.

VI. Conclusions

This paper addressed the problem of allocating a set of real-time tasks on a distributed system consisting of sensors nodes sending data to a coordinator, with the objective of satisfying feasibility constraints, while minimizing the overall energy consumption and maximizing task duplication.

Trade-offs between such two conflicting goals have been explored, by comparing two heuristic approaches against a complete method and simulated annealing. Simulation results show that one of the heuristic algorithms provides a reasonable solution, outperforming simulated annealing both in terms of efficiency and performance, especially for larger task set utilizations.

References

[1] N. Auluck and D. P. Agrawal. A scalable task duplication based algorithm for improving the schedulability of real-time heterogeneous multiprocessor systems. In *ICPP Workshops*, pages 89–96, 2003.

[2] H. Aydin and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, IPDPS '03, pages 113.2–, Washington, DC, USA, 2003. IEEE Computer Society.

[3] E. Bini and G. C. Buttazzo. Biasing effects in schedulability measures. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, Catania, Italy, June 2004.

[4] G. Buttazzo, E. Bini, and Y. Wu. Partitioning parallel applications on multiprocessor reservations. In *Proceedings of the 2010 22nd Euromicro Conference on Real-Time Systems*, ECRTS '10, pages 24–33, Washington, DC, USA, 2010. IEEE Computer Society.

[5] R. Davis and A. Burns. A survey of hard real-time scheduling algorithms and schedulability analysis techniques for multiprocessor systems. techreport YCS-2009-443, University of York, Department of Computer Science, 2009.

[6] I. Dietrich and F. Dressler. On the lifetime of wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(1):1 – 39, Feb 2009.

[7] J. Huang, C. Buckl, A. Raabe, and A. Knoll. Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems. In *PDP*, pages 447–454, 2011.

[8] A. Kandhalu, J. Kim, K. Lakshmanan, and R. R. Rajkumar. Energy-aware partitioned fixed-priority scheduling for chip multi-processors. *Real-Time Computing Systems and Applications, International Workshop on*, 1:93–102, 2011.

[9] V. Kianzad, S. S. Bhattacharyya, and G. Qu. Casper: An integrated energy-driven approach for task graph scheduling on distributed embedded systems. In *Proceedings of the 2005 IEEE International Conference on Application-Specific Systems, Architecture Processors*, ASAP '05, pages 191–197, Washington, DC, USA, 2005. IEEE Computer Society.

[10] J.-K. Kim, H. J. Siegel, A. A. Maciejewski, and R. Eigenmann. Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling. *IEEE Trans. Parallel Distrib. Syst.*, 19(11):1445–1457, 2008.

[11] M. Kim, S. Banerjee, N. Dutt, and N. Venkatasubramanian. Energy-aware cosynthesis of real-time multimedia applications on mpsocs using heterogeneous scheduling policies. *ACM Trans. Embed. Comput. Syst.*, 7(2):1–19, 2008.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[13] C. Li and L. Li. Energy constrained resource allocation optimization for mobile grids. *J. Parallel Distrib. Comput.*, 70(3):245–258, 2010.

[14] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[15] M. Lombardi and M. Milano. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17(1):51–85, January 2012.

[16] X. Qin and H. Jiang. A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems. *Parallel Comput.*, 32(5):331–356, June 2006.

[17] K. Ramamritham. Allocation and scheduling of precedence-related periodic tasks. *IEEE Trans. Parallel Distrib. Syst.*, 6(4):412–420, April 1995.

[18] M. Schmitz, B. Al-Hashimi, and P. Eles. Energy-efficient mapping and scheduling for dvs enabled distributed embedded systems. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '02, pages 514–, Washington, DC, USA, 2002. IEEE Computer Society.

[19] A. Schranzhofer, J.-J. Chen, L. Santinelli, and L. Thiele. Dynamic and adaptive allocation of applications on mpsoc platforms. In *Proc. of the 15th IEEE Conf. on Asia and South Pacific Design Automation Conference (ASP-DAC'10)*, pages 885 – 890, Taipei, Taiwan, Jan. 2010.

[20] K. Shin and D. Peng. *Static Allocation of Periodic Tasks with Precedence Constraints in Distributed Real-time Systems*. Technical report (International Computer Science Institute). International Computer Science Institute, 1988.

[21] T. Xie, X. Qin, and M. Nijim. Solving energy-latency dilemma: Task allocation for parallel applications in heterogeneous embedded systems. In *ICPP '06: Proceedings of the 2006 International Conference on Parallel Processing*, pages 12–22, Washington, DC, USA, 2006. IEEE Computer Society.

[22] C. J. Xue, Z. Yuan, G. Xing, Z. Shao, and E. Sha. Energy efficient operating mode assignment for real-time tasks in wireless embedded systems. In *Proceedings of the 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 237–246, Kaohsiung, Taiwan, August 2008.