Technical Report

Allocation of Parallel Real-Time Tasks in Distributed Multi-core Architectures supported by an FTT-SE Network

Ricardo Garibay-Martínez

Geoffrey Nelissen

Luis Lino Ferreira

Luis Miguel Pinho

CISTER-TR-150305

2015/03/24

Allocation of Parallel Real-Time Tasks in Distributed Multi-core Architectures supported by an FTT-SE Network

Ricardo Garibay-Martínez, Geoffrey Nelissen, Luis Lino Ferreira, Luis Miguel Pinho

CISTER Research Center

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: rgmaz@isep.ipp.pt, grrpn@isep.ipp.pt, llf@isep.ipp.pt, lmp@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

Distributed real-time systems such as automotive applications are becoming larger and more complex, thus, requiring the use of more powerful hardware and software architectures. Furthermore, those distributed applications commonly have stringent real-time constraints. This implies that such applications would gain in flexibility if they were parallelized and distributed over the system. In this paper, we consider the problem of allocating fixed-priority fork-join Parallel/Distributed real-time tasks onto distributed multi-core nodes connected through a Flexible Time Triggered Switched Ethernet network. We analyze the system requirements and present a set of formulations based on a constraint programming approach. Constraint programming allows us to express the relations between variables in the form of constraints. Our approach is guaranteed to find a feasible solution, if one exists, in contrast to other approaches based on heuristics. Furthermore, approaches based on constraint programming have shown to obtain solutions for these type of formulations in reasonable time.

Allocation of Parallel Real-Time Tasks in Distributed Multi-core Architectures supported by an FTT-SE Network

Ricardo Garibay-Martínez, Geoffrey Nelissen, Luis Lino Ferreira, and Luís Miguel Pinho

CISTER/INESC-TEC Research Centre, ISEP/IPP
Rua Dr. António Bernardino de Almeida 431, 4200-072 PORTO, Portugal
{rgmaz,grrpn,llf,lmp}@isep.ipp.pt

Abstract. Distributed real-time systems such as automotive applications are becoming larger and more complex, thus, requiring the use of more powerful hardware and software architectures. Furthermore, those distributed applications commonly have stringent real-time constraints. This implies that such applications would gain in flexibility if they were parallelized and distributed over the system. In this paper, we consider the problem of allocating *fixed-priority fork-join Parallel/Distributed real-time tasks* onto distributed multi-core nodes connected through a Flexible Time Triggered Switched Ethernet network. We analyze the system requirements and present a set of formulations based on a *constraint programming* approach. Constraint programming allows us to express the relations between variables in the form of constraints. Our approach is guaranteed to find a feasible solution, if one exists, in contrast to other approaches based on heuristics. Furthermore, approaches based on constraint programming have shown to obtain solutions for these type of formulations in reasonable time.

Keywords: Constraint Programming, Real-Time, Parallel Tasks, Distributed Multi-core Architectures

1 Introduction

Modern cars are a good example of time-constrained distributed systems. They are composed of tens of computing nodes, some of them based on multi-core architectures interconnected by various types of communication networks. The complexity of their workload never stops increasing, therefore, many of their applications would gain in flexibility if they were parallelized and distributed over the system.

The fork-join Parallel/Distributed real-time model (P/D tasks) [1], was designed to consider such execution pattern. In this paper, we consider P/D tasks and a distributed computing platform composed of multi-core nodes, and interconnected by a Flexible Time Triggered - Switched Ethernet (FTT-SE) network [2]. A P/D task starts with a master thread executing sequentially, which may then fork to be executed in parallel on local and remote nodes. When the parallel execution is completed on the local and remote nodes, the partial results are transmitted using messages, and aggregated by the master thread. The master thread then resumes its execution until the next fork. Since the threads are potentially distributed over the different nodes composing the platform, we call these operations Distributed-Fork (D-Fork) and Distributed-Join (D-Join).

Furthermore, for a given task set and a given computing platform, the main challenge is to find a feasible allocation for the tasks in a way that all the tasks meet their associated end-to-end deadlines. An end-to-end deadline represents the longest elapsed time that a sequence of threads and messages composing a task is permitted to take from the time instant at which it is activated, and the instant at which the last thread of the task completes its execution.

Contribution. In this paper, we present a set of formulations for modeling the allocation of P/D tasks in a distributed multi-core architecture by using a constraint programming approach. Constraint programming approach expresses the relations between variables in the form of constraints. Our constraint programming formulation is guaranteed to find a feasible allocation, if one exists, in contrast to other approaches based on heuristic techniques. Our work is close to the one presented in [4], but with the main difference: (i) that we model fork-join Parallel/Distributed real-time tasks executing over a distributed multi-core architecture, and (ii) that we consider messages being transmitted through a Flexible Time Triggered Switched Ethernet (FTT-SE) network. Furthermore, similar approaches based on constraint programming have shown that it is possible to obtain solutions for these type of formulations in reasonable time [3, 4].

Structure of the paper. Section 2 presents the related work. In Section 3 we introduce the system model. We introduce the constraint programming formulation in Section 4. Finally, our conclusions are drawn in Section 5.

2 Related work

In this section, we briefly review work related to: (i) scheduling of fixed-priority parallel real-time tasks, and (ii) the problem of allocating tasks and messages in distributed systems. Nevertheless, we restrain our attention to the case of real-time pre-emptive fixed-priority scheduling.

Research related to the scheduling of fixed-priority parallel real-time tasks has essentially targeted multi-core architectures. In [5], the authors introduced the *Task Stretch Transformation* (TST) model for parallel synchronous tasks that follow a fork-join structure. The TST considers preemptive fixed-priority periodic tasks with implicit deadlines partitioned according to the *Fisher-Baruah-Baker First-Fit-Decreasing* (FBB-FFD) [6] algorithm. Similarly, the *Segment Stretch Transformation* (SST) model was introduced in [7]. The authors converted the parallel threads of a fork-join task into sequential tasks by creating a master thread, but with the difference (when compared to [5]) that no thread is ever allowed to migrate between cores. That work was generalized in [8], by allowing an arbitrary number of threads per parallel segment, and in [9] for the scheduling of tasks represented by a *Directed Acyclic Graph* (DAG).

The problem of allocating sequential task in distributed systems has been intensively studied. Related works can be divided into: (i) heuristic based, and (ii) optimal strategies.

Related to heuristics based research, Tindell *et al.* [10] addressed these issues as an optimization problem, solving it with the general purpose Simulated Annealing algorithm. In [11] the authors assume a set of tasks and messages that are statically allocated to processors and networks (therefore no partitioning phase is considered), focusing on assigning the priorities to tasks and messages. Azketa *et al.* [12], addressed this problem by using the general purpose genetic algorithms. The authors initiate their genetic algorithm by assigning priorities using the HOPA heuristic [11], which is based on Deadline Monotonic (DM) priority assignment [13], and iterate over different solutions. To test schedulability they use the holistic analysis presented in Tindell *et al.* [14] and Palencia *et al.* [15, 16] schedulability tests. In [17] we proposed the DOPA heuristic, which simultaneously solves the problem of assigning tasks to processors and assigning priorities to tasks. DOPA is based on Audsleys Optimal Priority Assignment (OPA) algorithm [18] to assign priorities to tasks and messages.

Regarding optimal strategies, in [19] a solution based on branch-and-bound was proposed, enumerating the possible paths that can lead to an allocation, and cutting the path whenever a feasible schedule cannot be reached by following such task assignment. The bounding step is performed by checking the schedulability of each branch, based on the schedulability analysis derived by Tindell *et al.* [14]. In [3] the authors propose to solve the problem of allocation of tasks by formulating a mixed integer linear programming framework. Similarly to this work, in [4], the authors model the task partitioning problem as a constraint optimization programming problem. Both works assume that each thread has its own period and deadline.

In the previous work [1] we studied the problem of scheduling fork-join tasks on a distributed system composed of single-processor nodes and a shared bus communication network. Distributed systems have the particularity that the transmission delay of messages communicating threads within a task, cannot be deemed negligible as in the case of multi-core systems [5, 7, 8]. In here, we extend the problem of task allocation of fork-join real-time tasks presented in [1], by considering (i) a distributed multi-core architecture, and (ii) using a FTT-SE network for message transmission.

3 System Model

We consider a distributed computing platform composed of a set $N = \{\nu_1, \ldots, \nu_m\}$ of m multi-core nodes to execute tasks. Each node ν_r ($r \in \{1, \ldots, m\}$) is composed of m_r identical cores $\pi_{r,s}$ ($s \in \{1, \ldots, m_r\}$). The total number of cores in the system is therefore equal to $m_{tot} = \sum_{\nu_i \in N} m_r$. The processing nodes are interconnected by an FTT-SE network $\rho = \{SW_1, \ldots, SW_w\}$ of w Ethernet switches. The switches and distributed nodes are interconnected through full-duplex links.

Also, we consider a set $T = \{\tau_1, \ldots, \tau_n\}$ of n periodic P/D tasks. Figure 1 shows an example of a P/D task τ_i. A task τ_i is activated with a period T_i, and is characterized by an implicit end-to-end deadline D_i. A P/D task τ_i ($i \in \{1, \ldots, n\}$) is composed of a sequence of n_i sequential and parallel distributed segments $\sigma_{i,j}$ ($j \in \{1, \ldots, n_i\}$). n_i is assumed to be an odd integer, since a P/D task should always start and finish with a sequential segment. Therefore, odd segments $\sigma_{i,2j+1}$ identify sequential segments and even segments $\sigma_{i,2j}$ identify P/D segments. Each segment $\sigma_{i,j}$ is composed of a set of threads $\theta_{i,j,k}$ with $k \in \{1, \ldots, n_{i,j}\}$, where $n_{i,j} = 1$ for sequential segments.

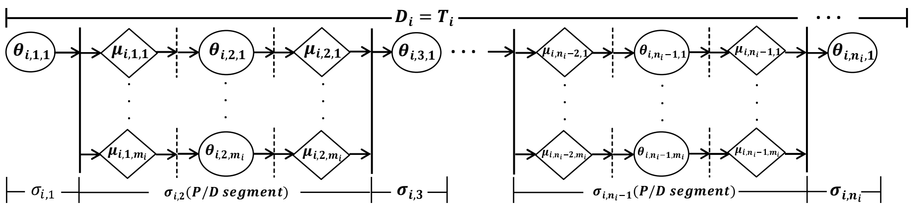

Fig. 1. The fork-join parallel distributed periodic real-time task (P/D task) model.

All sequential segments of a P/D task τ_i belong to the master thread, therefore, they are assumed to execute on the same core. This means that the core that performs a D-Fork operation (invoker core) is in charge of aggregating the result by performing a D-Join operation. Some threads within a P/D segment may be executed on remote node ν_l. Consequently, for each thread $\theta_{i,j,k}$ belonging to a P/D segment, two messages $\mu_{i,j-1,k}$ and $\mu_{i,j,k}$ are transmitted between the invoker and remote core. That is, P/D threads and messages that belong to a P/D segment and execute on a remote core, have a precedence relation: $\mu_{i,j-1,k} \rightarrow \theta_{i,j,k} \rightarrow \mu_{i,j,k}$. We call this sequence a *distributed execution path* (denoted as $DP_{i,j,k}$). If a P/D thread executes on the same node ν_l than the master thread, the transmission time of $\mu_{i,j-1,k}$ and $\mu_{i,j,k}$ are equal to zero, since the transfer of data through a shared memory can be considered negligible.

For each P/D segment, there exists a *synchronization point* at the end of the segment, indicating that no thread that belongs to the segment after the synchronization point can start executing before all threads of the current segment have completed their execution. Threads are preemptive, but messages are non-preemptive. Each thread $\theta_{i,j,k}$ has a Worst-Case Execution Time (WCET) of $C_{i,j,k}$, and each message $\mu_{i,j,k}$ has a Worst-Case Message Length (WCML) $M_{i,j,k}$.

4 Constraint Programming Formulation

The problem of task allocation can be seen as a two-sided problem: (i) finding the partitioning of threads and messages onto the processing elements of the distributed system, and (ii) finding the priority assignment for the threads and messages in that partition so that the real-time tasks and messages complete their execution before reaching their respective end-to-end deadlines.

In this section we analyze the system requirements and provide a formulation based on a constraint programming approach similar to [4].

4.1 Parallel/Distributed Tasks

In a similar manner as in [1], we transform threads composing a P/D task into a set of *independent sequential* tasks with constrained deadlines. This transformation is based on the imposition of a set of *artificial intermediate deadlines* (denoted as $\mathsf{d_{i,j}}$), to threads $\theta_{i,j,k}$ and messages $\mu_{i,j,k}$, in each segment $\sigma_{i,j}$. The following two constraints must be associated to each intermediate deadline $\mathsf{d_{i,j}}$.

Even if all threads execute in parallel, the relative deadline $\mathsf{d_{i,j}}$ cannot be smaller than the maximum WCET of a thread in that segment, thereby imposing that:

$$\bigwedge_{\forall \tau_i \in T} \bigwedge_{\forall \sigma_{i,j} \in \tau_i} \mathsf{d_{i,j}} \geq \max_{k=1,\ldots,n_{i,j}} \{C_{i,j,k}\}. \tag{1}$$

Also, the total execution granted to all segments constituting a task τ_i must be smaller or equal than the relative deadline of τ_i, that is:

$$\bigwedge_{\forall \tau_i \in T} \sum_{\forall \sigma_{i,j} \in \tau_i} \mathsf{d_{i,j}} \leq D_i. \tag{2}$$

Thus, the artificial deadline $\mathsf{d_{i,j}}$ is the maximum time that threads of a segment $\sigma_{i,j}$ are permitted to take, from the moment they are released, to the moment they complete their execution. Therefore, the problem can be formulated as to find the artificial deadlines $\mathsf{d_{i,j}}$ for every segment $\sigma_{i,j}$, in a way that the Worst-Case Response Time (WCRT) of threads $\theta_{i,j,k}$ (and messages $\mu_{i,j,k}$) is smaller or equal to the end-to-end deadline D_i. More constraints are presented in Sections 4.2 and 4.3.

4.2 Fully-Partitioned Distributed Multi-core Systems

In this work, we assume a fixed-priority fully-partitioned scheduling algorithm. Let us assume that each core in the system (regardless the processing node they are part of) is assigned a unique identifier in the interval $[1, m_{tot}]$. Then we define the integer variable $\Pi_{\theta_{\mathsf{i,j,k}}}$, indicating the identifier of the core on which the thread $\theta_{i,j,k}$ is mapped. By definition of the core identifier, the following constraints apply:

$$\Pi_{\theta_{\mathsf{i,j,k}}} > 0, \tag{3}$$

$$\Pi_{\theta_{\mathsf{i,j,k}}} \leq m_{tot}. \tag{4}$$

A constraint of the P/D task model is that all sequential segments of a task τ_i must execute on the same core $\pi_{r,s}$. This is imposed by (5):

$$\bigwedge_{\forall \theta_{i,2j+1,1} \in T} \ \bigwedge_{\forall \theta_{i,2b+1,1} \in T} \Pi_{\theta_{\mathsf{i,2j+1,1}}} = \Pi_{\theta_{\mathsf{i,2b+1,1}}}. \tag{5}$$

Let us define the variable $\mathsf{p_{i,j,k}}$ as the priority of a thread $\theta_{i,j,k}$. Although $\mathsf{p_{i,j,k}}$ could be an integer variable of the problem for which the solver finds a valid value in its proposed solution, in a concern of drastically reducing the number of variables and therefore the complexity of the problem, one may also assume that priorities are assigned using DM [13], in which case $\mathsf{p_{i,j,k}} = \mathsf{d_{i,j}}$, and $\mathsf{p_{i,j,k}}$ can be omitted in the description of the problem. Yet, it is necessary to evaluate if a certain partitioning leads to a valid solution. We know from [20], that the worst-case response time $\mathsf{r_{i,j,k}}$ of an independent thread $\theta_{i,j,k}$ scheduled with a preemptive fixed-priority scheduling algorithm, is given by (6):

$$\mathsf{r_{i,j,k}} = C_{i,j,k} + \sum_{\theta_{a,b,c} \in \mathsf{HP_{i,j,k}}} \left\lceil \frac{\mathsf{r_{i,j,k}}}{T_a} \right\rceil C_{a,b,c}, \tag{6}$$

where $\mathsf{HP_{i,j,k}}$ is the set of threads with higher or equal priority than $\theta_{i,j,k}$, and executing on the same core than $\theta_{i,j,k}$.

This can be modeled in the constraint problem as:

$$\bigwedge_{\forall \theta_{i,j,k} \in T} \mathsf{r_{i,j,k}} = C_{i,j,k} + \sum_{\forall \theta_{a,b,c} \in T} \mathsf{IHP^{a,b,c}_{i,j,k}}, \tag{7}$$

where $\mathsf{IHP^{a,b,c}_{i,j,k}}$ is the interference caused by a thread $\theta_{a,b,c}$ on $\theta_{i,j,k}$.

Higher priority relation is represented by the following boolean variable:

$$\mathsf{p^{a,b,c}_{i,j,k}} = \begin{cases} 1 & \text{if } \theta_{a,b,c} \text{ has higher priority than } \theta_{i,j,k} \ (\mathsf{p_{i,j,k}} \leq \mathsf{p_{a,b,c}}), \\ 0 & \text{otherwise.} \end{cases}$$

Because $\Pi_{\theta_{\mathsf{i,j,k}}} = \Pi_{\theta_{\mathsf{a,b,c}}}$ indicates that the $\theta_{i,j,k}$ and $\theta_{a,b,c}$ threads execute on the same core, the total interference over a thread $\theta_{i,j,k}$ is expressed as:

$$\bigwedge_{\forall \theta_{i,j,k} \in T} \ \bigwedge_{\forall \theta_{a,b,c} \in T} \mathsf{IHP^{a,b,c}_{i,j,k}} = \begin{cases} \mathsf{I^{a,b,c}_{i,j,k}} \times C_{a,b,c} & \text{if } \left((\mathsf{p^{a,b,c}_{i,j,k}} = 1) \wedge (\Pi_{\theta_{\mathsf{i,j,k}}} = \Pi_{\theta_{\mathsf{a,b,c}}}) \right), \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where $\mathsf{I^{a,b,c}_{i,j,k}}$ is the number of preemptions a thread $\theta_{i,j,k}$ suffers from a thread $\theta_{a,b,c}$. Since $\mathsf{I^{a,b,c}_{i,j,k}}$ is an integer, the ceiling operator can be rewritten as follows:

$$\left\lceil \frac{\mathsf{r_{i,j,k}}}{T_a} \right\rceil = \mathsf{I^{a,b,c}_{i,j,k}} \quad \Longrightarrow \quad \frac{\mathsf{r_{i,j,k}}}{T_a} \leq \mathsf{I^{a,b,c}_{i,j,k}} < \frac{\mathsf{r_{i,j,k}}}{T_a} + 1, \tag{9}$$

thereby, leading to the following constraints:

$$\bigwedge_{\forall\theta_{i,j,k}\in T}\ \bigwedge_{\forall\theta_{a,b,c}\in T} (\Pi_{\theta_{i,j,k}} = \Pi_{\theta_{a,b,c}}) \quad\rightarrow\quad (I^{a,b,c}_{i,j,k} \times T_a \geq r_{i,j,k}) \wedge \left((I^{a,b,c}_{i,j,k} - 1) \times T_a < r_{i,j,k}\right), \tag{10}$$

$$\bigwedge_{\forall\theta_{i,j,k}\in T}\ \bigwedge_{\forall\theta_{a,b,c}\in T} (\Pi_{\theta_{i,j,k}} \neq \Pi_{\theta_{i,j,k}}) \quad\rightarrow\quad I^{a,b,c}_{i,j,k} = 0. \tag{11}$$

Furthermore, in the P/D task model, some threads within a P/D segment may be executed on remote nodes. Consequently, for each such thread $\theta_{i,j,k}$, two messages $\mu_{i,j-1,k}$ and $\mu_{i,j,k}$ are transmitted between the invoker and remote node. That is, a distributed execution path is generated ($\mu_{i,j-1,k} \rightarrow \theta_{i,j,k} \rightarrow \mu_{i,j,k}$).

$\mathsf{NV}(\theta_{i,j,k})$ is a function denoting to which node ν_q a thread $\theta_{i,j,k}$ has been assigned. Then, $\mathsf{NV}(\theta_{i,j,k}) = \mathsf{NV}(\theta_{a,b,c})$ indicates that the threads $\theta_{i,j,k}$ and $\theta_{a,b,c}$ execute on the same node, in which case no message is transmitted through the network. However, if $\mathsf{NV}(\theta_{i,j,k}) \neq \mathsf{NV}(\theta_{a,b,c})$, the WCRT $r_{DP_{i,j,k}}$ of a distributed execution path $DP_{i,j,k}$ must be as follows:

$$\bigwedge_{\forall\mu_{i,j,k}\in T}\ \bigwedge_{\forall\theta_{i,j,k}\in T} r_{DP_{i,j,k}} = \begin{cases} r^{msg}_{i,j-1,k} + r_{i,j,k} + r^{msg}_{i,j,k} & \text{if } \mathsf{NV}(\theta_{i,j,k}) \neq \mathsf{NV}(\theta_{a,b,c}), \\ r_{i,j,k} & \text{otherwise,} \end{cases} \tag{12}$$

where $r_{i,j,k}$ is the WCRT of thread $\theta_{i,j,k}$ obtained with (7), and $r^{msg}_{i,j-1,k}$ and $r^{msg}_{i,j,k}$ are the WCRTs of messages $\mu_{i,j-1,k}$ and $\mu_{i,j,k}$ respectively, obtained with a network dependent analysis. In this paper, we assume the network analysis presented in [21] for FTT-SE networks. Thus, for a partition of tasks τ_i to be considered a valid solution (all deadlines are met), the following condition has to be respected:

$$\bigwedge_{\forall\theta_{i,j,k}\in T} r_{DP_{i,j,k}} \leq d_{i,j}. \tag{13}$$

4.3 FTT-SE Network

The communications within a FTT-SE network are done based on fixed duration slots called *Elementary Cycles* (ECs). The construction of the EC schedule is done by keeping updated tables for synchronous (i.e., periodic) and asynchronous (i.e., sporadic) messages. The scheduler applies a scheduling policy (e.g., Deadline Monotonic) over these tables, generating the ready queues for transmission for that EC. This process is repeated until no other message fits in its respective scheduling window for that EC (i.e., considering all messages from higher to lower priority). For building the ECs it is important to consider:

i. the architecture of the distributed system. The architectural model must include the full-duplex transmission links. We represent the architecture as an *adjacency-matrix* of a graph $G = (V, E)$. The set $V = \{v_1, \dots, v_{|V|}\}$ of vertices v_i represents the set of switches ρ and the set of nodes N, and the set $E = \{(v_1, v_2), \dots, (v_{|V|-1}, v_{|V|})\}$ of edges (v_i, v_j), represent the communication links, from nodes to switches, from switches to nodes or between switches. Note that: (i) direct links between nodes do not exist, (ii) links are directed; that is, (v_i, v_j) and (v_j, v_i) represent two different links, and (iii) the network is full-duplex; that is, if (v_i, v_j) is part of the graph, then (v_j, v_i) is too. Thus, the adjacency matrix representation of a graph G consists of a $|V| \times |V|$ matrix $A = (a_{i,j})$ such that:

$$a_{i,j} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

depending of the partitioning of threads onto the nodes ν_l of the system, there exists a set $\mathsf{PN}_{\mu_{\mathsf{i,j,k}}} \subseteq V$ containing the vertices (i.e., switches) that a message $\mu_{i,j,k}$ traverses during a D-fork or a D-join operation. For determining $\mathsf{PN}_{\mu_{\mathsf{i,j,k}}}$, we use the Breadth-First Search (BFS) Algorithm [22] for each message $\mu_{i,j,k}$. The BFS inputs are: the matrix A (representing the system architecture), the origin vertex (invoker core/remote core), and the destination vertex (the remote core/invoker core). The BFS finds the shortest path from the origin node to the destination node. Therefore, the BFS algorithm finds the switches that a message $\mu_{i,j,k}$ crosses during a D-fork or a D-join operation. The set $\mathsf{PN}_{\mu_{\mathsf{i,j,k}}}$ is required for computing the WCRT of a message $\mu_{i,j,k}$ in the FTT-SE network.

ii. the switching delays. In this paper, we consider a switching delay (denoted as $\mathsf{SD}_{\mathsf{i,j,k}}$) when a message $\mu_{i,j,k}$ crosses a switch SW_z. $\mathsf{SD}_{\mathsf{i,j,k}}$ has two components, the switch relaying latency (denoted as $\varDelta$), which has a constant value related to the specifications of the switch, and the Store-and-Forward Delay (denoted as $\mathsf{SFD}_{\mathsf{i,j,k}}$), i.e., $\mathsf{SD}_{\mathsf{i,j,k}} = \mathsf{SFD}_{\mathsf{i,j,k}} + \varDelta$. However, for each EC, only the maximum switching delay $\mathsf{SD}_{\mathsf{i,j,k}}$ is considered.

iii. the EC is subdivided into time slots for transmitting different types of traffic (e.g. synchronous window, asynchronous window, etc.). Thus, one must consider the length of the specific transmission window for each type of traffic (denoted as LW). The length of such a window is the reserved bandwidth for transmission in that EC, and cannot be exceeded when transmitting messages within the FTT-SE protocol. This is modeled by the *request bound function* in (14), and the *supply bound function* (19), presented in the following.

Response Time Analysis for FTT-SE networks. Depending on a given partition, we have to find the WCRT of the messages in the network to verify if the condition in (13) is respected. We consider the work presented in [21] for the computation of the WCRT of messages within the FTT-SE protocol, with a slight modification.

The **request bound function** $\mathsf{rbf}_{\mathsf{i,j,k}}(\mathsf{t})$ represents the maximum transmission requirements generated by a message $\mu_{i,j,k}$ and all its higher priority messages during an interval $[0, t]$. The $\mathsf{rbf}_{\mathsf{i,j,k}}(\mathsf{t})$ is computed as:

$$\bigwedge_{\forall \mu_{i,j,k} \in T} \mathsf{rbf}_{\mathsf{i,j,k}}(\mathsf{t}) = M_{i,j,k} + \mathsf{sn}_{\mathsf{i,j,k}} \times \mathsf{SFD}_{\mathsf{i,j,k}} + \mathsf{Wl}_{\mathsf{i,j,k}}(\mathsf{t}) + \mathsf{Wr}_{\mathsf{i,j,k}}(\mathsf{t}), \tag{14}$$

where, $\mathsf{sn}_{\mathsf{i,j,k}}$ is the number of switches that a message $\mu_{i,j,k}$ traverses from the origin node to its destination node, $\mathsf{Wl}_{\mathsf{i,j,k}}(\mathsf{t})$ is the *"Shared Link Delay"*, and $\mathsf{Wr}_{\mathsf{i,j,k}}(\mathsf{t})$ is the *"Remote Link Delay"*, which are explained below.

Shared Link Delay. The transmission of a message $\mu_{i,j,k}$ may be delayed by all the higher priority messages that share a link with $\mu_{i,j,k}$. However, such interference occurs only once, so messages that caused such interference on a previous link are excluded from the analysis for the next links. Also, when building the schedule for each EC, the scheduler considers the maximum switching delay SD_{z} (see (16)), only once. Therefore, $\mathsf{Wl}_{\mathsf{i,j,k}}(\mathsf{t})$ is computed by separating the interference of messages from the *switching-delay-effect* (denoted as $\mathsf{Is}_{\mathsf{i,j,k}}(\mathsf{t})$) for each EC. The shared link delay is computed in (15):

$$\mathsf{Wl}_{\mathsf{i,j,k}}(\mathsf{t}) = \sum_{\forall \mu_{a,b,c} \in \mathsf{SLD}_{\mathsf{i,j,k}}} \left\lceil \frac{t}{T_a} \right\rceil M_{a,b,c} + \mathsf{Is}_{\mathsf{i,j,k}}(\mathsf{t}), \tag{15}$$

where $\mathsf{SLD}_{\mathsf{i,j,k}} = \{\forall \mu_{a,b,c} : \mu_{a,b,c} \neq \mu_{i,j,k} \wedge (\mathsf{PN}_{\mu_{\mathsf{i,j,k}}} \cap \mathsf{PN}_{\mu_{\mathsf{a,b,c}}} \neq 0) \wedge \mu_{a,b,c} \in \mathsf{hp}(\mu_{\mathsf{i,j,k}}) \wedge \mu_{a,b,c} \in WT(\mu_{i,j,k})\}$, where, $\mathsf{hp}(\mu_{\mathsf{i,j,k}})$ is the set of messages with priority higher or equal than $\mu_{a,b,c}$ and $WT(\mu_{i,j,k})$ is the set of messages that are scheduled in the same window as $\mu_{a,b,c}$ (i.e. the synchronous or the asynchronous window). The set $\mathsf{hp}(\mu_{\mathsf{i,j,k}})$ for messages $\mu_{i,j,k}$ in (15), as well as the ceiling function, can be formulated in a similar manner as in Section 4.2.

For computing the switching-delay-effect $\mathsf{Is}_{\mathsf{i,j,k}}(\mathsf{t})$, it is needed to compute an upper bound on the number of switching delays ($\mathsf{SD}_{\mathsf{i,j,k}}$) from each message that contributes to (15), at time t. In [21], depending on time t, a number of switching delays are inserted into an array whenever a message crosses a switch in the network. The array is sorted in order to consider the maximum switching delays only. A sorting operation is not amenable to optimization solvers. Therefore, we introduce a simpler upper bound with the cost of slightly increment the pessimism.

The number of ECs in an interval $[0, t]$ is given by: $\mathsf{z}(\mathsf{t}) = \lceil \frac{t}{EC} \rceil$ (the ceiling function, can be formulated as in Section 4.2), thus, in order to consider the worst-case scenario for the computation of the WCRT, we consider the maximum switching delay ($\mathsf{SD}^{\max}_{\mathsf{i,j,k}}$) for each message that contributes to (15), and computed as:

$$\mathsf{SD}^{\max}_{\mathsf{i,j,k}} = \max_{\forall \mu_{a,b,c} \in \mathsf{SLD}_{\mathsf{i,j,k}}} \{\mathsf{SFD}_{\mathsf{i,j,k}} + \Delta\}. \tag{16}$$

Then, the maximum switching delay is multiplied by the number of ECs at time t (given by $\mathsf{z}(\mathsf{t})$). Thus, the switching-delay-effect is computed as:

$$\mathsf{Is}_{\mathsf{i,j,k}} = \mathsf{SD}^{\max}_{\mathsf{i,j,k}} \times \mathsf{z}(\mathsf{t}). \tag{17}$$

Remote Link Delay. A message $\mu_{i,j,k}$ can be blocked by other higher priority messages even if they do not share a transmission link. Thus, a higher priority message can delay a lower priority message even though they do not share a transmission link [21]. Therefore, to compute the worst-case remote link delay, it is needed to consider all messages that share links with the messages that contributed to the shared link delay (see (15)), excluding all messages that are already considered in (15). Hence, we have:

$$\mathsf{Wr_{i,j,k}(t)} = \sum_{\forall \mu_{p,q,r} \in \mathsf{RLD_{i,j,k}}} \left\lceil \frac{t}{T_p} \right\rceil M_{p,q,r} \tag{18}$$

where, $\mathsf{RLD_{i,j,k}} = \{\forall \mu_{p,q,r} : \mu_{p,q,r} \neq \mu_{a,b,c} \neq \mu_{i,j,k} \wedge (\mathsf{PN}_{\mu_{\mathsf{p,q,r}}} \cap \mathsf{PN}_{\mu_{\mathsf{a,b,c}}} \neq 0) \wedge (\mathsf{PN}_{\mu_{\mathsf{p,q,r}}} \cap \mathsf{PN}_{\mu_{\mathsf{i,j,k}}} = 0)(\mathsf{PN}_{\mu_{\mathsf{a,b,c}}} \cap \mathsf{PN}_{\mu_{\mathsf{i,j,k}}} \neq 0) \wedge \mu_{p,q,r} \in hp(\mu_{a,b,c}) \wedge \mu_{p,q,r} \in WT(\mu_{a,b,c})\}$.

The demand bound function is then compared with the **supply bound function** $\mathsf{sbf_{i,j,k}(t)}$, which represents the minimum effective communication capacity that the network supplies during the time interval $[0, t]$ to a message $\mu_{i,j,k}$. In each EC, the bandwidth provided for transmitting each type of traffic (e.g., synchronous or asynchronous traffic) is equal to $\frac{(LW-I)}{EC}$, where LW is an input and represents the length of the specific transmission window and I is the maximum inserted idle time of such window. The inserted idle time results from the fact that the maximum window duration cannot be exceeded.

$$\bigwedge_{\forall \mu_{i,j,k} \in T} \mathsf{sbf_{i,j,k}(t)} = (\frac{LW - I}{EC}) \times t. \tag{19}$$

Then, the response time of a message $\mu_{i,j,k}$ is computed by introducing a new variable $t_{i,j,k}$ such that:

$$\bigwedge_{\forall \mu_{i,j,k} \in T} \mathsf{t_{i,j,k}} > 0, \tag{20}$$

$$\bigwedge_{\forall \mu_{i,j,k} \in T} \mathsf{sbf_{i,j,k}(t_{i,j,k})} \geq \mathsf{rbf_{i,j,k}(t_{i,j,k})}. \tag{21}$$

Since it is not possible to determine the specific time of transmission of messages inside an EC, the computation of the WCRT for a message $\mu_{i,j,k}$ is in terms of a number of ECs, thus the WCRT of a message $\mu_{i,j,k}$ is given by:

$$\bigwedge_{\forall \mu_{i,j,k} \in T} \mathsf{r^{msg}_{i,j,k}} = \left\lceil \frac{\mathsf{t_{i,j,k}}}{EC} \right\rceil \times EC. \tag{22}$$

4.4 Constraint Satisfiability

The constraints sketched above are a combination of linear and non-linear constraints over a set of integer and boolean variables. This implies the use of extremely powerful optimization methods. It has been shown (e.g., [4]) that such type of optimization problems are not amenable for conventional numerical optimization solvers. However, for real-time purposes, a correct solution is obtained by guaranteeing that all the constraints are satisfied, regardless of the value of a given objective function. Thus, the optimization problem gets reduced to a Satisfiability (SAT) problem, in which solutions can be obtained in reasonable time [4]. The constrains and optimization variables are summarized in the following.

Summary. We convert a set of P/D tasks τ_i into a set of independent sequential tasks, by imposing a set of artificial intermediate deadlines. The constraints for intermediate deadline are: (1) and (2). A valid partition, in which all threads respect their intermediate deadlines $\mathsf{d}_{\mathsf{i,j}}$, is constrained with (5) and (7). The WCRT of a distributed execution path ($DP_{i,j,k}$) depends on where the threads in a P/D segment are executed (i.e., locally or remotely), that is modeled in (12). If threads $\theta_{i,j,k}$ are executed remotely, the WCRT of messages transmitted through an FTT-SE network has to be considered. That is modeled with (20)-(21). Finally, all tasks have to respect the condition in (13).

5 Conclusions

In this paper we presented the formulations for modeling the allocation of P/D tasks in a distributed multi-core architecture supported by an FTT-SE network, by using a constraint programming approach. Our constraint programming approach is guaranteed to find a feasible allocation, if one exists, in contrast to other approaches based on heuristic techniques. Furthermore, similar approaches based on constraint program have shown that it is possible to obtain solutions for these formulations in reasonable time.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within project FCOMP-01-0124-FEDER-037281 (CISTER); by FCT and the EU ARTEMIS JU funding, ARROWHEAD (ARTEMIS/0001/2012, JU grant nr. 332987), CONCERTO (ARTEMIS/0003/2012, JU grant nr. 333053); by FCT and ESF (European Social Fund) through POPH (Portuguese Human Potential Operational Program), under PhD grant SFRH/BD/71562/2010.

References

1. Garibay-Martínez, R.; Nelissen, G.; Ferreira, L.L.; Pinho, L.M.: On the scheduling of fork-join parallel/distributed real-time tasks. In: 9th IEEE International Symposium on Industrial Embedded Systems, pp. 31-40, (June 2014).
2. Marau, R., Almeida, L., Pedreiras, P.: Enhancing real-time communication over cots ethernet switches. In: IEEE International Workshop on Factory Communication Systems, pp. 295-302. (2006).
3. Zhu, Q., Zeng, H., Zheng, W., Natale, M. D., Sangiovanni-Vincentelli, A.: Optimization of task allocation and priority assignment in hard real-time distributed systems. ACM Transactions on Embedded Computing Systems, 11(4), 85. (2012).

4. Metzner, A., Herde, C.: Rtsat–an optimal and efficient approach to the task allocation problem in distributed architectures. In: 27th IEEE Real-Time Systems Symposium, pp. 147-158.(2006, December).
5. Lakshmanan, K., Kato, S., Rajkumar, R.: Scheduling parallel real-time tasks on multi-core processors. In: 31st IEEE Real-Time Systems Symposium pp. 259-268. (2010, November).
6. Fisher, N., Baruah, S., Baker, T. P.: The partitioned scheduling of sporadic tasks according to static-priorities. In: 18th Euromicro Conference on Real-Time Systems pp. 10-pp. (2006).
7. Fauberteau, F., Midonnet, S., Qamhieh, M.: Partitioned scheduling of parallel real-time tasks on multiprocessor systems. ACM SIGBED Review, 8(3), 28-31. (2011).
8. Saifullah, A., Li, J., Agrawal, K., Lu, C., Gill, C.: Multi-core real-time scheduling for generalized parallel task models. Real-Time Systems, 49(4), 404-435. (2013).
9. Qamhieh, M., George, L., Midonnet, S.: A Stretching Algorithm for Parallel Real-time DAG Tasks on Multiprocessor Systems. In: 22nd International Conference on Real-Time Networks and Systems (p. 13). (2014, October).
10. Tindell, K. W., Burns, A., Wellings, A. J.: Allocating hard real-time tasks: an NP-hard problem made easy. Real-Time Systems, 4(2), 145-165 (1992).
11. García, J. G., Harbour, M. G.: Optimized priority assignment for tasks and messages in distributed hard real-time systems. In: Third IEEE Workshop on Parallel and Distributed Real-Time Systems, pp. 124-132. (1995, April).
12. Azketa, E., Uribe, J. P., Gutirrez, J. J., Marcos, M., Almeida, L.: Permutational genetic algorithm for the optimized mapping and scheduling of tasks and messages in distributed real-time systems. In: 10th International Conference on Trust, Security and Privacy in Computing and Communications. (2011).
13. Leung, J. Y. T., Whitehead, J.: On the complexity of fixed-priority scheduling of periodic, real-time tasks. Performance evaluation, 2(4), 237-250 (1982).
14. Tindell, K., Clark, J.: Holistic schedulability analysis for distributed hard real-time systems. Microprocessing and microprogramming, 40(2), 117-134 (1994).
15. Palencia, J. C., Gonzalez Harbour, M.: Schedulability analysis for tasks with static and dynamic offsets. In: 19th IEEE Real-Time Systems Symposium, pp. 26-37. (1998, December).
16. Palencia, J. C., Gonzalez Harbour, M.: Exploiting precedence relations in the schedulability analysis of distributed real-time systems. In: 20th IEEE Real-Time Systems Symposium, pp. 328-339. (1999).
17. Garibay-Martínez, R., Nelissen G., Ferreira L. L., Pinho L. M.: Task partitioning and priority assignment for hard real-time distributed systems. In: International Workshop on Real-Time and Distributed Computing in Emerging Applications, (2013).
18. Audsley, N. C.: Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. University of York, Dep. of Computer Science. (1991).
19. Richard, M., Richard, P., Cottet, F.: Allocating and scheduling tasks in multiple fieldbus real-time systems. In: IEEE Conference on Emerging Technologies and Factory Automation, pp. 137-144. (2003, September).
20. Joseph, M., Pandya, P.: Finding response times in a real-time system. The Computer Journal, 29(5), 390-395. (1986).
21. Ashjaei, M., Behnam, M., Nolte, T., Almeida, L.: Performance analysis of master-slave multi-hop switched ethernet networks. In: 8th IEEE International Symposium Industrial Embedded Systems, pp. 280-289. (2013, June).
22. Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2001). Introduction to algorithms (Vol. 2, pp. 531-549). Cambridge: MIT press.