CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Conference Paper

Holistic Analysis for Fork-Join Distributed Tasks supported by the FTT-SE Protocol

Ricardo Garibay-Martínez*
Geoffrey Nelissen*
Luis Lino Ferreira*
Paulo Pedreiras
Luis Miguel Pinho*

*CISTER Research Center

CISTER-TR-150507

2015/05/27

Holistic Analysis for Fork-Join Distributed Tasks supported by the FTT-SE Protocol

Ricardo Garibay-Martínez*, Geoffrey Nelissen*, Luis Lino Ferreira*, Paulo Pedreiras, 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

This paper presents a holistic timing analysis for fixed-priority fork-join Parallel/Distributed tasks (P/D tasks) over a Flexible Time Triggered - Switched Ethernet (FTT-SE) network. The holistic approach considers both time-triggered and event-triggered tasks/messages.

Holistic Analysis for Fork-Join Distributed Tasks supported by the FTT-SE Protocol

Ricardo Garibay-Martínez*, Geoffrey Nelissen*, Luis Lino Ferreira*, Paulo Pedreiras†, Luís Miguel Pinho*
* CISTER/INESC-TEC Research Centre, ISEP
Polytechnic Institute of Porto, Porto, Portugal
{rgmaz, grrpn, llf, lmp}@isep.ipp.pt
† IEETA
University of Aveiro, Aveiro, Portugal
pbrp@ua.pt

***Abstract*—This paper presents a holistic timing analysis for fixed-priority fork-join Parallel/Distributed tasks (P/D tasks) over a Flexible Time Triggered - Switched Ethernet (FTT-SE) network. The holistic approach considers both time-triggered and event-triggered tasks/messages.**

I. Introduction

Modern automotive applications require the use of tens of interconnected Electronic Control Units (ECUs). But current network technologies only provide low bandwidth and the ECUs have relatively reduced processing power. In the future, applications will require higher bandwidth and more powerful resources (to accommodate for instance infotainment applications [1]). Future applications will require a network that conciliates real-time traffic guarantees (time-triggered and event-triggered), best-effort traffic, on-line scheduling, etc., and also they will require the use of more powerful computing models. For those reasons, in this work we propose the use of a Flexible Time Triggered - Switched Ethernet network (FTT-SE) [2], and the use of distributed real-time applications which are composed of a set of fork-join Parallel/Distributed real-time tasks (P/D tasks) [3].

Contribution. We propose a holistic timing analysis for the computation of the Worst-Case Response Time (WCRT) for P/D tasks when transformed by the Distributed Stretch Transformation (DST) algorithm [3], a technique allowing to reduce the number of messages transmitted through the network. We consider both synchronous and asynchronous communication patterns, considering an FTT-SE transmission network. Although not proved in this paper, the presented analysis is not limited to the DST algorithm and can be used in any distributed system, using an FTT-SE network, scheduled with a fixed-priority algorithm.

II. System Model

We consider a distributed computing platform composed of a set $\pi = \{\pi_1, \ldots, \pi_m\}$ of m identical uni-core processors. The processors are interconnected by an FTT-SE network ρ. ρ is composed of a set $\{SW_1, \ldots, SW_r\}$ of r Ethernet switches. The switches SW_x ($x \in \{1, \ldots, r\}$), and their respective links interconnect all the distributed nodes in the network.

We consider a set $\tau = \{\tau_1, \ldots, \tau_n\}$ of n sporadic P/D tasks (see Fig. 1). A task τ_i is activated with a minimum inter-arrival time T_i and is characterized by an implicit end-to-end deadline D_i. τ_i is composed of a sequence of n_i sequential and parallel distributed segments $\sigma_{i,j}$ ($j \in \{1, \ldots, n_i\}$). That is, a P/D task should always start and finish with a sequential segment. Odd segments $\sigma_{i,2j+1}$ identify sequential segments and even segments $\sigma_{i,2j}$ identify P/D segments. The number of segments n_i is assumed to be an odd integer. 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 and $n_{i,j} = m_i \leq m$ for P/D segments.

All sequential segments of a P/D task τ_i must execute on the same processor. This means that the processor that performs a Distributed-Fork (D-Fork) operation (the invoker node) is in charge of aggregating the results by performing a Distributed-Join (D-Join) operation. Threads within a P/D segment may be executed on remote nodes. Consequently, for each thread $\theta_{i,2j,k}$ belonging to a P/D segment, two messages $\mu_{i,2j-1,k}$ and $\mu_{i,2j,k}$ are transmitted between the invoker and remote node. That is, P/D threads and messages that belong to a P/D segment and execute on a remote node have a precedence relation: $\mu_{i,2j-1,k} \rightarrow \theta_{i,2j,k} \rightarrow \mu_{i,2j,k}$. We call this sequence a *distributed execution path* (denoted as $DP_{i,2j,k}$).

For each P/D segment, there exists a *synchronization point* at the end of each segment, meaning that no thread that belongs to the segment after the synchronization point can start executing before: (i) all threads of the current segment have completed their execution, and (ii) all messages have been received.

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) $M_{i,j,k}$. Threads are preemptive, but messages are non-preemptive.

III. The FTT-SE Protocol

The FFT-SE protocol makes use of the master/slave paradigm [2], where a dedicated node (the master node) schedules messages on the network. Communications within a FTT-SE network are done based on fixed duration time slots, called *Elementary Cycles* (ECs). For building the message schedule in an EC, one must consider: (i) the characteristics of the transmission links; (ii) the length of the specific transmission window for each type of traffic (e.g., synchronous or asynchronous); and (iii) the multiple switching delays in the network: when transmitting messages with FTT-SE, a switching delay (denoted as $SD^{\max}_{i,j,k}$) must be considered when a message $\mu_{i,j,k}$ crosses a subset of the switches in ρ (denoted as $R_{i,j,k}$). We consider that the switching delay has two components, the switch relaying latency (denoted as Δ), and the Store-and-Forward Delay of a message $\mu_{i,j,k}$ (denoted as $SFD_{i,j,k}$), i.e.,

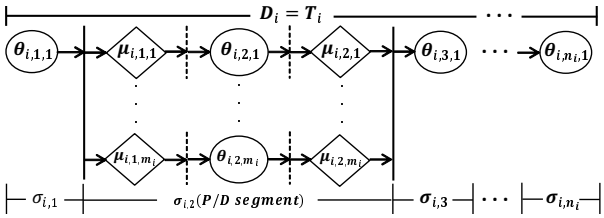

Fig. 1: The fork-join parallel distributed real-time task (P/D task) model.

$SD^{\max}_{i,j,k} = \max_{\mu_{a,b,c} \in \mathsf{CUR(EC)}}\{SFD_{a,b,c}+\Delta\}$, where $\mathsf{CUR(EC)}$ represents the set of messages transmitted in the current EC.

A. Worst-Case Response Time in FTT-SE networks

In this section we review the work presented in [4] for the computation of the WCRT of messages within the FTT-SE protocol. For notational convenience, we replaced the original notation, with the notation introduced in Section II.

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

$$rbf_{i,j,k}(t) \stackrel{\text{def}}{=} M_{i,j,k} + sn_{i,j,k} \times SD^{\max}_{i,j,k} + Wl_{i,j,k}(t) + Wr_{i,j,k}(t), \quad (1)$$

where, $sn_{i,j,k}$ is the number of switches that $\mu_{i,j,k}$ traverses from the origin node to its destination (nodes in $R_{i,j,k}$), $Wl_{i,j,k}(t)$ is the *shared link delay*, and $Wr_{i,j,k}(t)$ is the *remote link delay*. The Shared Link Delay and the Remote Link delay are briefly explained below. For further details, please refer to [4].

The *shared link delay* considers the delay caused by messages sharing the same transmission links as $\mu_{i,j,k}$. $Wl_{i,j,k}(t)$ is computed by separating the interference of messages from the *switching-delay-effect* (denoted as $Is_{i,j,k}(t)$) for each EC. Which is given by:

$$Wl_{i,j,k}(t) \stackrel{\text{def}}{=} \sum_{\forall \mu_{a,b,c} \in SLD_{i,j,k}} \left\lceil \frac{t}{T_a} \right\rceil M_{a,b,c} + Is_{i,j,k}(t), \quad (2)$$

where $SLD_{i,j,k} = \{\mu_{a,b,c} \in \tau \mid \mu_{a,b,c} \neq \mu_{i,j,k} \wedge R_{i,j,k} \cap R_{a,b,c} \neq \emptyset \wedge \mu_{i,j,k} \in hp(\mu_{i,j,k}) \wedge \mu_{i,j,k} \in WT(\mu_{i,j,k})\}$, where, $hp(\mu_{i,j,k})$ is the set of messages with priority higher than $\mu_{i,j,k}$ and $WT(\mu_{i,j,k})$ is the set of messages that are scheduled in the same window as $\mu_{i,j,k}$. Details on how to compute the switching-delay-effect $Is_{i,j,k}(t)$, can be found in [4].

A message $\mu_{i,j,k}$ can be blocked by other higher priority messages even if they do not share a transmission link. This is considered by the *remote link delay* given below:

$$Wr_{i,j,k}(t) \stackrel{\text{def}}{=} \sum_{\forall \mu_{p,q,r} \in RLD_{i,j,k}} \left\lceil \frac{t}{T_p} \right\rceil M_{p,q,r}, \quad (3)$$

where $RLD_{i,j,k} = \{\mu_{p,q,r} \in \tau \mid \mu_{p,q,r} \neq \mu_{a,b,c} \neq \mu_{i,j,k} \wedge R_{d,e,f} \cap R_{a,b,c} \neq \emptyset \wedge R_{p,q,r} \cap R_{i,j,k} = \emptyset \wedge R_{p,q,r} \cap R_{i,j,k} \neq 0 \ \wedge \mu_{p,q,r} \in hp(\mu_{a,b,c}) \wedge \mu_{p,q,r} \in WT(\mu_{a,b,c})\}$.

The demand bound function is then compared with the *supply bound function* $sbf_{i,j,k}(t)$, which represents the minimum effective communication capacity that the network supplies during the time interval $[0,t]$. Thus $sbf_{i,j,k}(t)$ is computed as:

$$sbf_{i,j,k}(t) \stackrel{\text{def}}{=} (\frac{LW - I}{EC}) \times t, \quad (4)$$

where LW is the length of the specific transmission window and I is the maximum inserted idle time of such a window.

Then, the response time of a message $\mu_{i,j,k}$ is computed by determining the time instant t^* such that:

$$t^* = \min(t > 0) : sbf_{i,j,k}(t) \geq rbf_{i,j,k}(t). \quad (5)$$

Thus the WCRT in number of ECs (denoted as $\mathsf{WR}(\mu_{i,j,k})$) of a message $\mu_{i,j,k}$, in a synchronous system is given by:

$$\mathsf{WR}^{syn}(\mu_{i,j,k}) = \left\lceil \frac{t^*}{EC} \right\rceil. \quad (6)$$

The previous analysis considers the transmission of synchronous messages. Details regarding the transmission of asynchronous messages are given in Section V-1.

IV. The DST Algorithm

The purpose of the DST algorithm [3] is to minimize the number of threads that execute in a remote node while respecting their deadlines. Thus, the DST reduces the number of messages transmitted through the network. To achieve that, the DST opts for the formation of a stretched master thread $\tau_i^{stretched}$ for each P/D task τ_i. The objective is to keep as many threads as possible for local execution, thus, reducing the load in the network and on the remote processors.

Example 1. *Consider* 2 *tasks* τ_1 *and* τ_2 *to be stretched by the DST. Task* τ_1 *is composed of a sequential thread* $\theta_{1,1,1}$ *with a WCET =* 1 *followed by P/D* 3 *threads* $\theta_{1,2,1-3}$ *with a WCET =* 2 *and* 1 *sequential thread* $\theta_{1,3,1}$ *with a WCET =* 1*; its deadline is equal to* 8*. Task* τ_2 *is composed of a sequential thread* $\theta_{2,1,1}$ *with a WCET =* 1 *followed by* 3 *P/D threads* $\theta_{2,2,1-3}$ *with a WCET =* 3 *and* 1 *sequential thread* $\theta_{2,3,1}$ *with a WCET =* 1*; its deadline is equal to* 10*. The DST first calculates the maximum execution length, which is given by* $C_i = \sum_j \sum_k C_{i,j,k}$ *[3]. For tasks* τ_1 *and* τ_2*, we obtain* $C_1 = 8$ *and* $C_2 = 11$*. Then, two cases have to be considered:*

1) $C_i \leq D_i$*. This is the case of* τ_1*; whenever such a case appears for a task* τ_i*,* τ_i *is fully stretched into a single thread and handled as a sequential task with execution time* C_i*, period* T_i*, and implicit deadline* D_i*. Therefore, no messages are exchanged through the network.*
2) $C_i > D_i$*. This is the case of* τ_2*; for such tasks, the DST inserts (coalesces) as many P/D threads of* τ_i *as possible into the master thread. To do so, it calculates the available slack (i. e.,* $L_i = D_i - \sum_j C_{i,j,1}$*), and the task capacity (i. e.,* $f_i = \frac{L_i}{\sum_{j=1}^{\frac{n_i-1}{2}} \max_k\{C_{i,2j,k}\}}$*) of* τ_i*. For* τ_2*, it gives* $L_2 = 10 - 5 = 5$ *and,* $f_2 = \frac{5}{3}$*. Thus, the number of P/D threads that each P/D segment can fully insert into the master thread without causing* τ_i *to miss its deadline is given by* $i_{i,2j} = \lfloor f_i \rfloor$*. In the case of* τ_2*,* $i_{2,2} = \lfloor f_2 \rfloor = 1$*.*

The number $q_{i,2j}$ of the remaining P/D threads $\theta_{i,j,k}$ that have not been coalesced into the master thread is given by: $q_{i,2j} = m_{i,2j} - i_{i,2j}$. The slack f_i of task τ_i is equally distributed between all the P/D segments of a P/D task τ_i.

Therefore, at the end of the DST transformation, a P/D task τ_i will be composed of: (i) a single sequential fully stretched task, or (ii) a single stretched master thread $\tau_i^{stretched}$ and a set of non-coalesced P/D threads. The stretched master thread $\tau_i^{stretched}$ is assigned to its own processor. The remaining single fully stretched sequential tasks and non-coalesced P/D threads are assigned according to any fixed-priority partitioning algorithm. Their respective messages are scheduled accordingly.

V. A Holistic Analysis for Stretched Tasks

In distributed systems, the impact of messages used for communication purposes cannot be deemed negligible as in the case of multiprocessor systems. In this section, we present a holistic analysis that assumes the impact of such messages on the WCRT. It is considered that the P/D tasks have been stretched using the DST transformation. We consider two types of communication patterns: time-triggered and event-triggered.

1) Node queuing delay: In the FTT-SE, messages are transmitted in periodic time windows called ECs. Thus, if a thread completes its execution just after the beginning of an EC, it has to wait for the beginning of the next EC in order to initiate the transmission of a message. We call this delay the *node queuing delay*. In the worst case it has a length of 1 EC. The node queuing delay has to be considered whenever a transmission is initiated by threads of a P/D task (i.e., during each D-fork and D-join operation). This means that Eq. (6) must be incremented by 1 EC for synchronous messages, and incremented by 3 ECs for the case of asynchronous messages (2 ECs due to the *signalling mechanism* overhead inherent to asynchronous messages in FTT-SE [4], and 1 due to the node queuing delay).

2) Time-triggered systems: When the activation of P/D messages is strictly periodic, it implies a time-triggered communication pattern, in which synchronous messages are used. For time-triggered systems, an offset indicates the earliest moment at which a thread $\theta_{i,j,k}$ (or message $\mu_{i,j,k}$) of a segment $\sigma_{i,j}$ can start its execution (or transmission, respectively). This offset is equal to the WCRT $\mathsf{WR}(\mu_{i,j-1,k})$ (resp., $\mathsf{WR}(\theta_{i,j,k})$) of the message (resp., thread) preceding $\theta_{i,j,k}$ (resp., $\mu_{i,j,k}$) in the fork-join task, thereby ensuring that the tasks and messages never experience any release jitter.

Two cases must be considered when computing the response time of a parallel task stretched with the DST transformation:

Fully stretched tasks: if a task has been fully stretched, no message must be sent over the network (Case 1 in Section IV). Therefore, its WCRT only depends on the suffered interference caused by other higher priority threads executing on the same processor. This can be computed by using the usual response time analysis for fixed-priority tasks [5]:

$$\mathsf{WR}(\tau_i) = C_i + \sum_{\forall \theta_{p,q,r} \in hp(\tau_i)} \left\lceil \frac{\mathsf{WR}(\tau_i) + J(\theta_{p,q,r})}{T_p} \right\rceil C_{p,q,r}, \quad (7)$$

where $hp(\tau_i)$ is the set of threads with higher priority than τ_i that execute on the same processor than τ_i, and $J(\theta_{p,q,r})$ being the maximum jitter on the arrival of $\theta_{p,q,r}$. This jitter is equal to 0 in time-triggered systems. Eq. (7) can be solved with a fixed point iteration over $\mathsf{WR}(\tau_i)$, where $\mathsf{WR}(\tau_i)$ is initialised at C_i for the first iteration.

Non-fully stretched tasks: for non-fully stretched tasks, one must consider the sequential and parallel segments independently. Remember that for each P/D segment, there exists a synchronization point, indicating that no thread that belongs to the segment after the synchronisation point can start executing before all threads of the current segment have completed their execution and the associated messages completed their transmission. Therefore, the WCRT of a task τ_i is computed based on the sum of the $\mathsf{WR}(\sigma_{i,j})$ of each segment $\sigma_{i,j}$:

$$\mathsf{WR}(\tau_i) = \textstyle\sum_{j=1}^{n_i} (\mathsf{WR}(\sigma_{i,j})), \quad (8)$$

where $\mathsf{WR}(\sigma_{i,j})$ can be computed as described below for sequential and parallel segments, respectively.

- *Sequential segments*. Sequential segments are executed on their own processors. Therefore, they do not suffer any interference from other threads. Hence:

$$\mathsf{WR}(\sigma_{i,2j+1}) = C_{i,2j+1,1}. \quad (9)$$

- *Parallel segments*. For a parallel segment $\sigma_{i,2j}$, the WCRT is given by the maximum of the two following values:
  - the sum of the WCETs of the set of threads coalesced in $\tau_i^{stretched}$ (denoted by $CThr_{i,2j}$), which are executed sequentially on their own processor. That is,

$$\mathsf{WR}(CThr_{i,2j}) = \textstyle\sum_{\theta_{i,2j,k} \in \{\sigma_{i,2j} \cap \tau_i^{stretched}\}} C_{i,2j,k}. \quad (10)$$

  - the maximum WCRT of each distributed execution paths $DP_{i,2j,k}$ within the parallel segment (denoted as $\mathsf{WR}^{\max}_{DP_{i,2j}}$). The WCRT of a distributed execution path $DP_{i,2j,k}$ is upper-bounded by the sum of the WCRT of its constituting messages $\mu_{i,2j-1,k}$ and $\mu_{i,2j,k}$, and its thread $\theta_{i,j,k}$, i.e.,

$$\mathsf{WR}(DP_{i,2j,k}) = \mathsf{WR}(\mu_{i,2j-1,k}) + \mathsf{WR}(\theta_{i,j,k}) + \mathsf{WR}(\mu_{i,2j,k}). \quad (11)$$

    Under the FTT-SE protocol, $\mathsf{WR}(\mu_{i,2j-1,k})$ and $\mathsf{WR}(\mu_{i,2j,k})$ can be computed using Eq. (6) increased by 1 EC (see V-1), and Eq. (12) gives the WCRT of $\theta_{i,j,k}$:

$$\mathsf{WR}(\theta_{i,j,k}) = C_{i,j,k} + \sum_{\forall \theta_{p,q,r} \in hp(\theta_{i,j,k})} \left\lceil \frac{\mathsf{WR}(\theta_{i,j,k}) + J(\theta_{p,q,r})}{T_p} \right\rceil C_{p,q,r}. \quad (12)$$

  Therefore, the maximum WCRT experienced by a distributed execution path $\sigma_{i,2j}$ is:

$$\mathsf{WR}^{\max}_{DP_{i,2j}} = \max_{DP_{i,2,j,k} \in \sigma_{i,2j}} \{\mathsf{WR}(DP_{i,2j,k})\}. \quad (13)$$

  Thus, the WCRT of a parallel segment $\sigma_{i,2j}$ is given by:

$$\mathsf{WR}(\sigma_{i,2j}) = \max\{\mathsf{WR}(CThr_{i,2j}), \mathsf{WR}^{\max}_{DP_{i,2j}}\}. \quad (14)$$

3) Event-triggered systems: The event-triggered communication pattern makes use of asynchronous messages. In an event-trigger system, threads and messages are sent on completion of the previous message (or thread) in the fork-join sequence, implying that each thread and message may experience a release jitter $J(\mu_{i,j,k})$ and $J(\theta_{i,j,k})$ respectively, equal to the difference between the Best-Case Response Time (BCRT) and the WCET of the preceding message (or thread, respectively). As shown by Eq. (7) and (12), these jitters have an impact on the WCET of the threads. The same is true for messages. Hence, we adapt Eq. (6) to consider their release jitter. Note that Eq. (7)–(14) remain unchanged.

Only the computation of $\mathsf{WR}(\mu_{i,2j-1,k})$ and $\mathsf{WR}(\mu_{i,2j,k})$ are altered by the release jitters. In fact, using the same reasoning than in [5], it is easy to prove that a message $\mu_{p,q,r}$ with a release jitter $J(\mu_{p,q,r})$ and interfering with $\mu_{i,j,k}$ may release at most $\lceil \frac{t+J(\mu_{p,q,r})}{T_p} \rceil$ message instances in a time window of length t. Therefore, Eq. (2) and (3) must be modified as follows:

$$Wl_{i,j,k}(t) = \sum_{\forall \mu_{p,q,r} \in SLD_{i,j,k}} \left\lceil \frac{t+J(\mu_{p,q,r})}{T_p} \right\rceil M_{p,q,r} + Is_{i,j,k}(t), \quad (15)$$

$$Wr_{i,j,k}(t) = \sum_{\forall \mu_{p,q,r} \in RLD_{i,j,k}} \left\lceil \frac{t+J(\mu_{p,q,r})}{T_p} \right\rceil M_{p,q,r}. \quad (16)$$

Thus, by assuming the BCRT equal to zero for all preceding events, $J(\theta_{i,j,k})$ ($J(\mu_{i,j,k})$, resp.) is equal to the largest sum of the WCRT of each predecessor, computed by Eq. (7)–(16), i.e:

$$J(\theta_{i,j,k}) = \max_{\forall \mu_{p,q,r} \in predec(\theta_{i,j,k})} \{J(\mu_{p,q,r}) + WCRT(\mu_{p,q,r})\}, \quad (17)$$

$$J(\mu_{p,q,r}) = \max_{\forall \theta_{i,j,k} \in predec(\mu_{p,q,r})} \{J(\theta_{i,j,k}) + WCRT(\theta_{i,j,k})\}, \quad (18)$$

where, $predec(\theta_{i,j,k})$ ($predec(\mu_{i,j,k})$) is the set of all threads (messages, resp.) which are direct predecessors of thread $\mu_{i,j,k}$ (message $\theta_{i,j,k}$, resp.) in the P/D task τ_i.

VI. Towards an Improved Response Time Analysis for Distributed Execution Paths

In this section, we provide intuitions on how the WCRT analysis for distributed execution paths can be improved. Consider the following example:

Example 2. *Consider a set of tasks stretched with the DST transformation and mapped onto processors by an arbitrary partitioning algorithm. Also, consider the system architecture depicted in Fig. 2. If a message $\mu_{i,j,k}$ is transmitted from the ECU Head-Unit (H-U) to CTRL-2, it has to cross two links in the network; from H-U to SW_1, and from SW_1 to CTRL-2. This is shown in Fig. 3. Let us assume that two threads $\theta_{1,2,8}$ and $\theta_{1,2,9}$ of task τ_1 are assigned to processor CTRL-2, thus, τ_1 sends two messages $\mu_{1,1,8}$ and $\mu_{1,1,9}$ from H-U to CTRL-2. After their remote execution is completed, both threads perform a D-Join operation sending the corresponding messages to their invoker node. Because the transmission of $\mu_{1,1,8}$ and $\mu_{1,1,9}$ occurs sequentially, the transmission of message $\mu_{1,1,9}$ is occurring in parallel with the beginning of the the execution of the first thread $\theta_{1,2,8}$. We denote this execution overlap as $\mathsf{OvF}^{\pi_i}_{l^d_{SW_x}}$. Similarly the transmissions of messages $\mu_{1,2,8}$ and $\mu_{1,2,9}$ during the D-join operation do not interfere with each other on the uplink $l^u_{\pi_2,SW_1}$. We denote this non-interference effect as $\mathsf{OvJ}^{l^d_{SW_x}}_{\pi_i}(\mu_{i,j,k})$.*

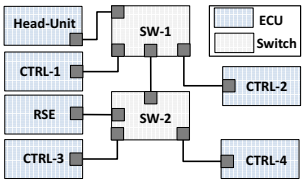

Fig. 2: Automotive architecture with an FTT-SE network.

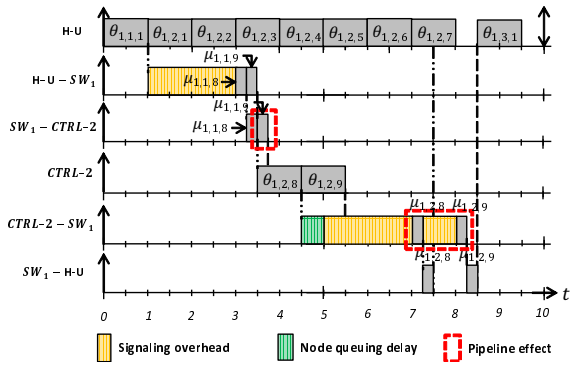

Fig. 3: Pipeline effect of a P/D task on an FTT-SE network.

This example illustrates that Eq. (11) is pessimistic when computing the WCRT of a distributed execution path. If an overlap $\mathsf{OvF}^{\pi_i}_{l^d_{SW_x}}$ and a non-interference effect $\mathsf{OvJ}^{l^d_{SW_x}}_{\pi_i}(\mu_{i,j,k})$ exist, it could be subtracted from the WCRT of a distributed execution path (Eq. (11)), thereby reducing the pessimism.

VII. Conclusions

In this paper, we presented a holistic timing analysis for the computation of the WCRT of P/D tasks when transformed by the DST algorithm and supported by an FTT-SE network. Our holistic approach considers the impact of messages used for communication/synchronization purposes that cannot be deemed negligible as in the case of multiprocessor systems. The analysis considers both synchronous and asynchronous communication patterns.

Acknowledgment: This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and when applicable, co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within project UID/CEC/04234/2013 (CISTER Research Centre); also by FCT/MEC and ERDF through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within project FCOMP-01-0124-FEDER-020447 (REGAIN); by FCT/MEC and the EU ARTEMIS JU within project ARTEMIS/0003/2012 - JU grant nr. 333053 (CONCERTO) and ARTEMIS/0001/2012 - JU grant nr. 332987 (ARROWHEAD); by FCT/MEC within project Serv-CPS (PTDC/EAA-AUT/122362/2010) ; by FCT/MEC and ESF (European Social Fund) through POPH (Portuguese Human Potential Operational Program), under PhD grant SFRH/BD/71562/2010.

References

[1] H.-T. Lim, L. Volker, and D. Herrscher, "Challenges in a future ip/ethernet-based in-car network for real-time applications," in *DAC'11*, June 2011, pp. 7–12.

[2] P. Pedreiras, P. Gai, L. Almeida, and G. Buttazzo, "Ftt-ethernet: a flexible real-time communication protocol that supports dynamic qos management on ethernet-based systems," *Industrial Informatics, IEEE Transactions on*, vol. 1, no. 3, pp. 162–172, Aug 2005.

[3] R. Garibay-Martínez, G. Nelissen, L. Ferreira, and L. Pinho, "On the scheduling of fork-join parallel/distributed real-time tasks," in *SIES'14*, June 2014, pp. 31–40.

[4] M. Ashjaei, M. Behnam, T. Nolte, and L. Almeida, "Performance analysis of master-slave multi-hop switched ethernet networks," in *SIES'13*, June 2013, pp. 280–289.

[5] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Software Engineering Journal*, vol. 8, no. 5, pp. 284–292, Sep 1993.