ECTM: A network-on-chip communication model to combine task and message schedulability analysis

Mourad Dridi, Frank Singhoff, Stéphane Rubini, Jean-Philippe Diguet

ECTM: A Network-on-Chip Communication Model to Combine Task and Message Schedulability Analysis

Mourad Dridi[a,*], Frank Singhoff[a], Stéphane Rubini[a], Jean-Philippe Diguet[b]

[a] *Univ. Brest, Lab-STICC, CNRS, UMR 6285, F-29200 Brest, France*
[b] *Univ. Bretagne Sud, Lab-STICC, CNRS, UMR 6285, F-56100 Lorient, France*

Abstract

Network-on-Chips (NoC) are widely used in industrial applications since they provide communication parallelism and reduce energy consumption. The use of NoC has been recently extended to real-time systems, whose execution has to meet temporal constraints. Communication delays introduced by the network make the scheduling analysis challenging. In this article, we propose a new NoC communication model called ECTM. The main goal of this model is to assess the schedulability of dependent periodic tasks exchanging messages on a NoC. ECTM is a model allowing schedulability analysis of messages and tasks of the NoC. To achieve schedulability, ECTM produces an analysis model by transforming NoC messages to tasks in order to take into account communication delays during the scheduling analysis. Schedulability of the system is assessed using simulation over the feasibility interval with a list scheduling, ECTM supports Store-And-Forward and Wormhole NoC.

In this article, we have demonstrated the correctness of the transformations of ECTM. ECTM has been implemented in a real-time scheduling analysis tool called Cheddar and we performed experiments to assess its efficiency. ECTM is more efficient than existing solutions with an improvement of 30% for Store-And-Forward NoCs and up to 100% for Wormhole NoCs, while the proposed model requires a larger computation time about 17% for Store-And-Forward NoCs

Keywords: Network-On-Chip (NoC), Real-Time Systems, Communication, Schedulability Analysis, Wormhole, Store-And-Forward

1. Introduction

Since they provide communication parallelism and limit the energy consumption, Networks-on-Chip (NoC) are widely used in industrial applications.

*Mourad Dridi
Email address: `Mourad.Dridi@univ-brest.fr` (Mourad Dridi)

Recently, the use of NoC [1] has been extended to real-time systems. Real-time systems are systems that have deadlines to meet [2].

NoCs introduce communication delays because of possible resource contentions between different flows in the network. Those delays depend on many factors. Some of them are related to the NoC configuration like the switching mode or the arbitration policy, while others result from the network state [3]. Thus, due to the NoC architecture, real-time scheduling analysis is a challenge.

In order to analyze schedulability of periodic tasks in NoC-based parallel architectures, we have to take into account at the same time the task scheduling over processing elements and the message communication scheduling over the network.

Unfortunately, classic multiprocessor real-time scheduling solutions consider worst case communication time instead of the actual delays introduced by the network [4]. Consequently, it leads to pessimistic schedulability analysis results.

In this article, we propose a NoC communication model in order to assess the scheduling analysis of periodic tasks over NoC architectures. The proposed model simplifies combined scheduling analysis of the periodic tasks and the NoC communications.

Our approach converts NoC flows of messages scheduling to periodic tasks scheduling. Each flow of message is transformed into a set of dependent periodic tasks. As a result, the scheduling analysis is simplified as the problem of the tasks and NoC communications scheduling is similar to the scheduling of periodic tasks deployed on a multiprocessor, which has been well studied [5, 6]. We have proven the correctness of the transformations. Furthermore, by a set of experiments, we show the efficiency of this approach.

Unlike conventional solutions which consider pessimistic scenarios to evaluate the communication times, ECTM only accounts real traffic contentions in the network. In fact, ECTM selects interference which arrives due to the actual flows in the network thanks to a temporal characterization of the traffic. For that, ECTM is intended to be more accurate than the state-of-the-art solutions for a given class of NoC architecture. More precisely, we can only apply our proposition to Virtual Channel Store-And-Forward (SAF) NoC and Private Virtual Channel Wormhole NoC.

The remainder of the article is organized as follows. The next section presents related works. Section 3 introduces background about the NoCs and the periodic task/flow models. Assumptions about the NoC architectures supported in this work are defined in the section 4. Then, the next section describes our approach for analyzing the scheduling of periodic tasks over such NoC-based parallel architectures. Section 5 demonstrates the correctness of the transformation of our analysis model. Implementation and evaluation of the proposed approach are explained in section 7 and section 8 concludes this article.

2. Related Work

NoC-based communications have recently attracted significant attention because of their potential for performance improvement and their impact on task scheduling. Thus, several NoC communication analysis methods have been proposed.

Shi and Burns [7] propose an analysis approach for real-time on chip communications with Wormhole switching and fixed priority scheduling. In [8], the authors focus on real-time communication services with a priority share policy. These works established worst case communication time analysis for different NoC configurations.

In order to schedule periodic tasks over NoC architectures, several scheduling algorithms have been proposed. Those solutions have different goals. Some optimize the design by minimizing the power consumption and the execution time of an application. In other works, the timing constraints of the system are enforced [9].

Varatkar et al. [10] developed a two-step mapping and scheduling algorithm. It performs simultaneous mapping and scheduling of tasks in order to reduce communication energy by minimizing the inter-processor communication. However, the communication distance is roughly approximated. This work does not carry out communication mapping and scheduling. Thus, real-time schedulability analysis cannot assess application real-time constraints, because communication latencies are completely ignored in the analysis process.

Lei et al. [11, 12] also propose a two-step algorithm for task mapping and scheduling over NoC architectures. The goal of the scheduling analysis is to check hard deadlines, while the goal of task mapping is to maximize timing performances. The communications are not considered in the mapping and the scheduling process. Communication delays are estimated using average distance in the NoC. Thus, this approach cannot guarantee hard deadlines required by critical real-time systems also.

The analysis approach we propose in this article starts with a model transformation. The model transformation aims to convert an architectural model into a simplified analysis model [13].

A model transformation approach has been used by Lakshmanan et al. in [14] also. They propose a stretching algorithm for a dependent task model. The stretching algorithm avoids the parallel structure of the architectural model by executing them as sequentially as possible. The work in [13] introduces a Directed Acyclic Graph (DAG) Stretching algorithm in order to analyze the scheduling of dependent tasks. In the stretching algorithms, dependent tasks are transformed to a set of independent sequential threads. Intermediate offsets and deadlines are assigned to threads so as to determine their execution interval. However, all these transformation models do not consider the communications in the NoC. Again, such an approach does not allow us to assess application real-time constraints.

To conclude, there are few methods that perform NoC communication scheduling analysis [9]. Furthermore, most of the proposed approaches for task and communication scheduling over NoC architectures do not consider the exact communication time. They consider worst case communication time or average distance in the NoC to estimate the communication time. Finally, existing transformation models also ignore the NoC communications.

The next sections detail a new NoC communication model that addresses the limitations highlighted above.

3. Background

This section sums up the background required to understand the contributions proposed in the next sections. First, we introduce the main concepts about NoCs. Then, we present the models of flow and task assumed in this article.

3.1. Network-on-Chip

A NoC is a network of nodes. Each node may be a processing element, a memory, a peripheral or a cluster.

Fig. 1 illustrates the major components of a NoC. Nodes access the network through a network interface (NI) and receive data encapsulated in packets [1]. Then, processing elements communicate by exchanging messages over the network. The network is composed of routers and unidirectional physical links. We note e_{RxRy} the unidirectional link between the two routers R_x and R_y, and e_{PExRy} the unidirectional link between the processing element PE_x and the router R_y. Finally, e_{RyPEx} is an unidirectional link between the router R_y and the processing element PE_x.

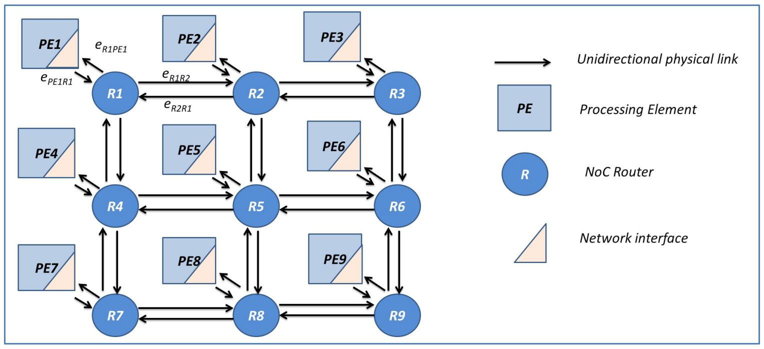

Figure 1: Network On Chip

In order to ensure the communication in the network, routers implement arbitration policy and switching mode.

Switching Mode: It determines how a packet is allocated to buffers and channels and when it will receive service. In this article, we assume two switching modes: Store And Forward (SAF) and Wormhole.

For the SAF mode, each router waits for a full packet to arrive before sending it to the next router [15].

With the Wormhole mode, the packet is divided into a number of fixed size flits [3]. More precisely, the packet is split into an header flit, one or several body flits and a tail flit. The header flit stores the routing information and is used to build the route. As the header flit moves ahead along the selected path, the remaining flits follow in a pipeline way and possibly span over multiple routers.

Arbitration Policy: Its aim is to select one packet among many in a router. When several incoming packets request the same output port of a router, an arbitration is required to select one of these packets.

Several arbitration mechanisms have been used in NoC routers [1]. Round-robin and priority-based are 2 examples of these mechanisms. Round-robin arbiters give the lowest priority to the last served request in the next arbitration. Priority-based arbiters choose one packet from many requests based on their fixed priority.

3.2. Task model

In this article, we consider real-time systems designed as a set of dependent periodic tasks deployed on a NoC.

We assume a set of tasks Γ composed of n periodic tasks: $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$ Each task τ_i is defined as follows: $\tau_i = \{\ O_{\tau_i},\ T_{\tau_i},\ C_{\tau_i},\ D_{\tau_i},\ \Pi_{\tau_i},\ Node_{\tau_i},\ E\ \}$ where:

- O_{τ_i} is the first release time of the task τ_i.
- T_{τ_i} is the period of the task.
- C_{τ_i} specifies the computation time needed by the task defined as its Worst Case Execution Time.
- D_{τ_i} is the deadline to meet.
- Π_{τ_i} is the fixed priority of the task. The value 1 denotes the highest priority level while a larger value is a lower priority.
- $Node_{\tau_i}$ identifies the NoC node (i.e. the processing element) running the task. This parameter allows us to introduce the mapping configuration, i.e. to which processing element each task is assigned to.
- $E : \Gamma \longrightarrow \Gamma^p$

  $\tau_i \longrightarrow E(\tau_i) = \{\tau_j, \dots, \tau_k\ \}$

  The function E introduces the precedence constraints of the task model. For a given task τ_i, the function E determines all the tasks $\{\tau_j, \dots, \tau_k\}$ that receive messages from the task τ_i.

From the previous task model, in [16], we have proposed DTFM (Dual Task and Flow Model) in order to compute the flow model from the task model, the mapping of the tasks on the processing elements and the NoC model. The flow model specifies the messages that have to be transmitted in the NoC to make effective the communications between the tasks.

With DTFM, the set of flows ψ related to Γ can be defined as follow. The flow model comprises m periodic traffic flows $\psi = \{\rho_1, \rho_2, \dots, \rho_m\}$. Each flow

ρ_i raises a sequence of messages in a similar way that a task raises a sequence of jobs. Each flow ρ_i is defined as follows: ρ_i ={ O_{ρ_i}, T_{ρ_i}, D_{ρ_i}, Π_{ρ_i}, $NodeS_{\rho_i}$, $NodeD_{\rho_i}$, F }

where:

- O_{ρ_i} is the release time of the messages, i.e. the first time a message of the flow becomes ready to be transmitted.
- T_{ρ_i} is the period of the message.
- D_{ρ_i} is the deadline of the message i.e. the message must be available for the receiver before this time.
- Π_{ρ_i} is the priority of the messages. The value 1 denotes the highest priority level while a larger value indicates a lower priority.
- $NodeS_{\rho_i}$ is the node (i.e. the processing element) running the transmitter task.
- $NodeD_{\rho_i}$ is the node running the receiver task.
- $F : \psi \longrightarrow \Omega^p$

  $\rho_i \longrightarrow F(\rho_i) = \{e_{PE_jR_j}, \ldots, e_{R_xR_y}, \ldots, e_{R_kPE_k}\}$

  where Ω is a set of physical links in the NoC.

  F is a function that computes the links used by a flow. In other words, F identifies the physical links that will be used by any messages of the flow.

  The function F helps us to understand the relationships between the various flows that transit through the network and to determine the interference between the messages sent by the tasks.

Finally, in this article, we assume the following protocol about message exchanges : tasks send their packets at completion time (at the end) of each instance of their periodic execution and the packets are read before the periodic execution of the receiving tasks. This protocol is called *immediate data connection protocol* in the standard AADL [17].

3.3. NoC communication analysis

Depending on the NoC configuration, we may have different types of latencies. In this section, we focus on latencies introduced by the NoC assumed in this article. Let's see the different types of delays that we must understand to assess schedulability of real-time tasks deployed on such NoCs.

3.3.1. Path delay

The Path Delay PD is the time required to send a packet from the transmitter to the receiver when no traffic flow contention exists.

We assume here that packets of a flow do not share any physical link with other flows along the path. This delay is variable and depends on several parameters such as the number of hops between the transmitter and the receiver, the flow rate and the size of the messages [3].

3.3.2. Delay due to Direct Interference

A second kind of delay occurs due to direct interference.

We have a direct interference when two flows share the same physical link at the same time. This delay caused by a direct interference between flows is called a Direct Delay DD.

The direct interference can be presented as a non-deterministic source of delay. In our analysis, we consider the worst case latency introduced by the direct interference. We assume the packet from the observed traffic-flow arrives just after other packets for each shared link. We assume also a maximum size of the packets [3].

3.3.3. Delay due to Indirect Interference

Another kind of delay occurs due to indirect interference. There is an indirect interference when two flows do not share the same physical link but interfere directly with the same flows. We call the delay caused by an indirect interference, an Indirect Delay, noted ID.

Considering indirect interference, each flow blocked by direct interference with other flows, will in turn block all other flows which share physical links with it. However those physical links remain available and cannot be used by any flows during the indirect interference situation. Thus, this kind of interference must be avoided because it affects significantly the use rate of the network resources. In addition, in most cases, indirect interference complicates the scheduling analysis and increases the pessimism degree of the worst case communication time.

In the next section, we detail our contributions and the targeted NoC configurations.

4. Supported NoC Architectures

In this section, we present assumptions about the NoC architectures investigated in this paper. Then, we discuss the rationale of these choices.

- **Assumptions 1:** We consider 2D NoC with XY/YX routing algorithm. We take this assumptions in order to consider the most used topology and routing algorithm in NoCs.
- **Assumptions 2:** We consider Wormhole or SAF as switching mode in the considered NoC router. Wormhole switching minimizes the communication time and it does not require large capacity buffers. But, we also consider SAF switching mode because it decreases the potential congestion over the network[18].
- **Assumptions 3:** We consider NoC router with virtual channels. This assumptions has been taken in order to reduce blocking situations between flows and to give more flexibility to flows in the considered network.

- **Assumptions 4:** We consider a fixed priority flit-level arbitration for Wormhole NoC and a fixed priority packet-level for SAF NoC. We take this assumptions since most of NoC routers use those arbitration policies.

- **Assumptions 5:** In this assumptions, we consider that the NoC router can communicate only one flow per virtual channel. Said differently, two flows cannot share the same virtual channel even if they can share the same physical link using two different virtual channels. We consider this assumption in order to avoid indirect interference situations.

To sum up, we target in our work the 2D mesh NoCs with several virtual channels. For SAF NoC, using virtual channel allows us to avoid indirect interference. With a Wormhole NoC, we need to use a virtual channel for each communication flow in order to also prevent such an interference [18]. Next, we demonstrate how those assumptions avoid indirect interference. Then, we discuss the strengths of those assumptions.

We prove by contradiction that we cannot have indirect interference in a SAF NoC which implements virtual channel. Let assume a SAF NoC with virtual channels. In this configuration, each virtual channel is only used by one flow. Then, two flows cannot share the same virtual channel.

Let consider three flows ρ_1, ρ_2 and ρ_3 as shown in Fig 2.

- ρ_1 and ρ_2 share the physical link e_{R1R2}.

- ρ_2 and ρ_3 share the physical link e_{R3R4}.

Now, let assume that we have indirect interference, that's mean the flow ρ_1 is blocked by the flow ρ_2 because they share the same physical link e_{R1R2} at the same time. To have an indirect interference, the flow ρ_2 must be also blocked by the flow ρ_3 because they share the same physical link e_{R3R4}. However, since the link e_{R1R2} is not used by either ρ_2 or ρ_1, they cannot share the same virtual. We can do the same demonstration for Wormhole NoC with virtual channel.

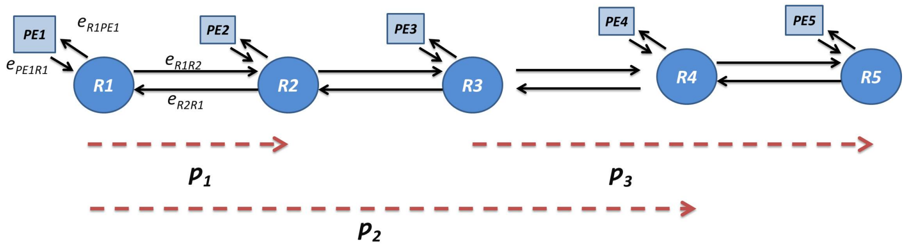

Figure 2: Indirect interferences in NoC architectures

Assumptions (3 and 5) allow us to avoid indirect interference in our analysis. Maybe the weakest assumption in our study is the restricted number of flow per virtual channel, i.e. only one flow per virtual channel (Assumptions 6). In order to enforce this assumption we would note that there are many solutions to mitigate its negative effects.

For M flows per physical link and under those assumptions, we must have M virtual channels per physical link. The value of M can be reduced depending to several parameters such as the task model and the task mapping. On the one hand, for a fixed value of M, we must select a task mapping which allows us to have at most M flows sharing the same physical link. This kind of problems is similar to a Quadratic Assignment Problem (QAP) [19]. QAP is known as an NP-Hard optimization problem. However, there are various heuristics that can be used for defining such mapping as those described in [19] or [20]. Multi-criteria heuristics are presented in these articles. Authors in [20, 19] try to optimize the total communication volume and the number of VCs. We note that this optimization issue is orthogonal to the scope of this work. On the other hand, for a fixed task mapping, we can select M as the maximum of flows sharing the same link. We note that the higher the number of virtual channel, the larger the overhead of area for the NoC implementation.

5. NoC Communication Time Models

In order to analyze the scheduling of periodic tasks running on a NoC architecture, communication delays in the NoC have to be investigated.

Shi et al. in [3, 7, 8] proposed NoC communication models based on worst case scenarios. Those communication models compute worst case communication delays that can be used to assess schedulability of periodic tasks running on the NoC.

Next, we formalize WCCTM, the Worst Case Communication Time Model, base on the Shi's analysis. Then, we propose ECTM (Exact Communication Time Model), an exact communication model that improves WCCTM.

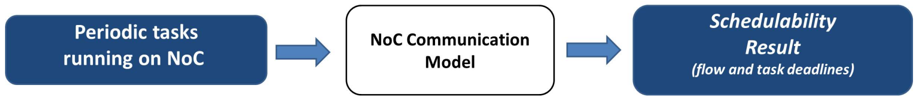

Figure 3: NoC Communication Model - Goal

As shown in Fig. 3, the goal of a NoC communication model is to assess schedulability of periodic tasks running on the NoC. Next, we present an overview of our schedulability approach before introducing both WCCTM and ECTM.

5.1. Overview of our approach

Fig. 4 shows an overview of the approach we propose. The system to analyze is expressed by a model of the NoC, a model of the application as a set of dependent periodic tasks and how such tasks are deployed on the processing elements. In the sequel, we call this input model the *architectural model*.

Then, to achieve schedulability analysis of the overall system, we apply the following steps:

1. The communication delay model of the architectural model is transformed into an *analysis model* to validate the scheduling of the communications. For such a transformation, we use ECTM and WCCTM. WCCTM is used in this article to evaluate ECTM efficiency.
2. By a scheduling simulation with a list scheduling policy over the feasibility interval, we assess the schedulability of the system. Feasibility intervals are choosen according to [21].

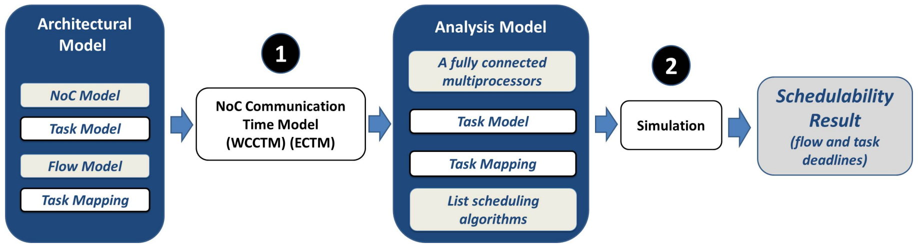

Figure 4: Overall approach

In the next paragraphs, we define the architectural model and the analysis model.

5.1.1. Architectural model

For the architectural model, we consider a set of periodic tasks exchanging messages, and deployed over a NoC. The task set is defined by the parameters introduced in Section 3.2.

The NoC is characterized by its topology, the switching mode and arbitration policy it implements. Table 1 summarizes the assumptions related to the NoC, i.e. mainly a 2D mesh NoC with virtual channels. We recall here that allocating a virtual channel for each flow allows us to avoid indirect interference.

Moreover, a task mapping completes the architecture model, that allows to apply the DTFM [16] algorithm to generate a temporal specification of the communication flows inside the NoC.

5.1.2. Analysis model

The analysis model is composed of a set of periodic tasks running on a multiprocessor execution platform. We assume a multiprocessor with identical processors [21] and no hardware shared resource.

Scheduling analysis of such models can be performed with list scheduling [22, 6]. List scheduling algorithms build scheduling list of tasks when assigning them their priorities. There are several means to determine the priorities of tasks such as Highest Level First, Longest Path and Longest Processing Time [23].

Noc Communication Model	WCCTM	$ECTM_{SAF}$	$ECTM_{Wormhole}$
Topology and dimension	2D mesh	2D mesh	2D mesh
Routing algorithm	XY / YX	XY / YX	XY / YX
Switching mode	Wormhole / SAF	SAF	Wormhole
Arbitration policy	Fixed priority Round Robin	Fixed priority Round Robin	Fixed priority Round Robin
Virtual channel	With	With	With
Preemption level	Flit / Packet	Packet	Flit

Table 1: Assumptions on the NoC for each communication model

Highest Level First with Estimated Time (HLFET), Modified Critical Path (MCP), Earliest Time First (ETF) and Dynamic Level Scheduling (DLS) are examples of list scheduling algorithms which can be applied on the analysis model we assume [5, 6].

Now we describe the communication time models considered in this article: WCCTM and two ECTM models, one for SAF switching mode and the second for Wormhole switching mode.

5.2. Worst Case Communication Time Model (WCCTM)

Let us see now how the architectural model is transformed into a WCCTM analysis model. To achieve this transformation, we apply the following rules:

- **Rule 1:** All routers and unidirectional links of the architectural model will be removed in the analysis model while keeping all the processing elements.
- **Rule 2:** We keep all tasks of the architectural model in the analysis model.
- **Rule 3:** Each flow ρ_i of the architectural model is modeled as a processing element PE_{ρ_i} in the analysis model. The tasks running in the new processing element PE_{ρ_i} are scheduled according to a fixed priority policy.
- **Rule 4:** Each flow ρ_i of the architectural model will be abstracted by one task τ_{ρ_i} in the analysis model.
- **Rule 5:** If the architectural model contains two periodic tasks τ_{source} and $\tau_{destination}$ and if τ_{source} sends the messages of the flow ρ_i to $\tau_{destination}$, then applying WCCTM leads to the flow ρ_i transformed in the analysis model as a periodic task τ_{ρ_i} with the following parameters:
  - $O_{\tau_{\rho_i}} = O_{\rho_i}$
  - $T_{\tau_{\rho_i}} = T_{\rho_i}$

- $C_{\tau_{\rho_i}} = WCCT_i$
  where $WCCT_i$ is the Worst Case Communication Time of a ρ_i's message. We compute $WCCT_i$ with the methods proposed in [3, 7, 8] for Wormhole NoCs or in [3] for SAF NoCs.
- $D_{\tau_{\rho_i}} = D_{\rho_i}$
- $Node_{\tau_{\rho_i}} = PE_{\rho_i}$
- $E(\tau_{\rho_i}) = \tau_{destination}$

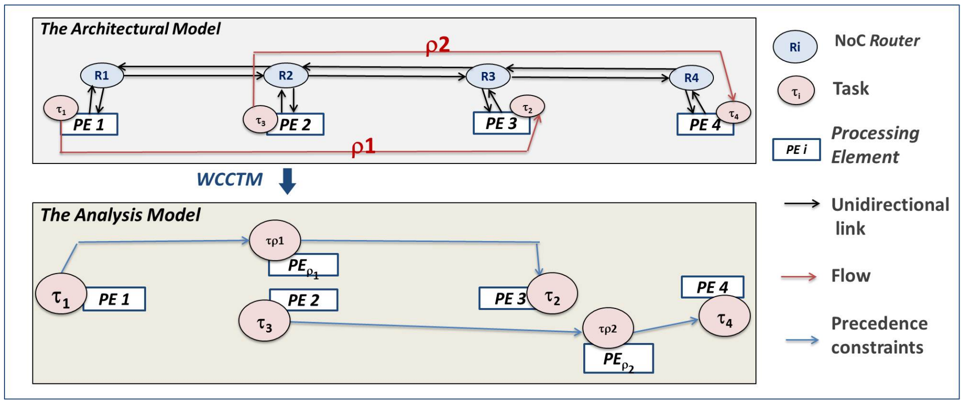

Figure 5: Worst Case Communication Time Model (WCCTM) - Example

We illustrate the WCCTM transformation by the example shown in Fig. 5. We consider 4 periodic tasks τ_1, τ_2, τ_3 and τ_4. τ_1 sends messages of the flow ρ_1 to the task τ_2, and τ_3 sends messages of the flow ρ_2 to the task τ_4.

In addition, the tasks τ_1, τ_2, τ_3 and τ_4 are respectively executed by the processing elements PE_1, PE_3, PE_2 and PE_4.

When applying the WCCTM transformation rules, we get:

- (**Rule 1**) Routers and links of the architectural model are removed in the analysis model.
- (**Rule 2 + Rule 1**) All tasks and processing elements of the architectural model are kept in the analysis model.
- (**Rule 3 + Rule 4**) The flow ρ_1 is transformed into a task τ_{ρ_1}. The processing element PE_{ρ_1} executes τ_{ρ_1}. The flow ρ_2 is transformed into a task τ_{ρ_2}. Last, the processing element PE_{ρ_2} executes τ_{ρ_2}.

In order to assess the schedulability of such system, we run scheduling simulation over the feasibility interval. In this simulation, we consider a list scheduling algorithm for processing elements PE_1, PE_2, PE_3 and PE_4. For the rest of processing elements, we consider a fixed priority algorithm.

In the architectural model, the flows ρ_1 and ρ_2 share the same physical link e_{R2R3}, while, in the analysis model, there is no shared resources between τ_{ρ_1} and τ_{ρ_2}. We explain that by considering the worst case communication time of each flow as the capacity of each corresponding task.

5.3. Exact Communication Time Model for SAF NoC

For the SAF NoCs, we propose $ECTM_{SAF}$. Next, we detail the transformation rules for $ECTM_{SAF}$:

- **Rule 1:** Each router of the architectural model will be removed in the analysis model while keeping all the processing elements.
- **Rule 2:** Each unidirectional link e_{RxRy} in the network between two routers Rx and Ry of the architectural model will be replaced in the analysis model by a new processing element PE_{RxRy}.

  Each unidirectional link e_{PExRx} in the network between a processing element and a router (respectively e_{RxPEx} between a router and a processing element) of the architectural model will be replaced in the analysis model by a new processing element PE_{PExRx} (respectively PE_{RxPEx}).

  These new processing elements in the analysis model apply a scheduling algorithm in agreement with the considered arbitration policy in the NoC router.
- **Rule 3:** We keep all tasks of the architectural model in the analysis model.
- **Rule 4:** Each flow ρ_i of the architectural model will be replaced by a set of $nbrlink_i$ tasks Γ_{ρ_i}, where $nbrlink_i$ denotes the number of links used by the flow ρ_i, or:

  $\Gamma_{\rho_i} = \{\ \tau_{\rho_{i,1}}, \tau_{\rho_{i,2}}, \dots, \tau_{\rho_{i,nbrlink_i}}\}$.
- **Rule 5:** If the architectural model contains two periodic tasks τ_{source} and $\tau_{destination}$ and if τ_{source} sends messages from the flow ρ_i to $\tau_{destination}$, then applying $ECTM_{SAF}$ leads to the flow ρ_i transformed in the analysis model as a set of periodic tasks Γ_{ρ_i}. Parameters of each task $\tau_{\rho_{i,j}}$ of the task set Γ_{ρ_i} are computed as follow.

  For j in $[\ 1, \dots nbrlink_i\ ]$ $\tau_{\rho_{i,j}}$ is characterized by :

  - $O_{\tau_{\rho_{i,j}}} = O_{\rho_i}$
  - $T_{\tau_{\rho_{i,j}}} = T_{\rho_i}$
  - $C_{\tau_{\rho_{i,j}}} = PD_{OneLink}$

    $PD_{OneLink}$ is the Path Delay of one link, i.e. the time required to send one flow over one link without considering the possible conflicts in the network.
  - $D_{\tau_{\rho_{i,j}}} = D_{\rho_i}$

– $Node_{\tau_{\rho_{i,j}}} = \begin{cases} PE_{R_j PE_j} & if\ j = nbrlink_i \\ PE_{PE_j R_j} & if\ j = 1 \\ PE_{R_x R_y} & if\ 1 < j < nbrlink_i \end{cases}$

We note here that e_{RxRy} represents each of the physical links used by the flow ρ_i which is transformed to the processing element $PE_{R_x R_y}$ in the analysis model.

– $E(\tau_{\rho_{i,j}}) = \begin{cases} \tau_{i,j+1} & if\ j < nbrlink_i \\ \tau_{destination} & if\ j = nbrlink_i \end{cases}$

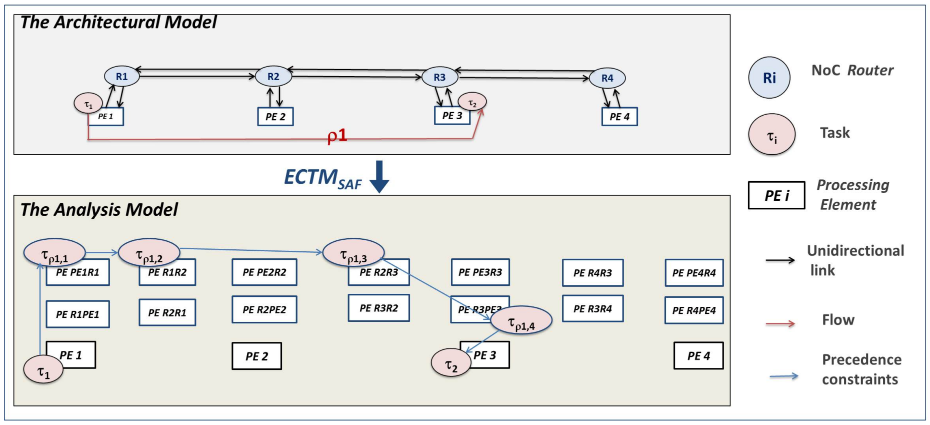

Figure 6: Example of $ECTM_{SAF}$ for one flow

We apply the $ECTM_{SAF}$ transformation to the previous example (Fig. 5). Figures 6 and 7 show the architectural and the analysis models with $ECTM_{SAF}$. Fig. 6 shows the transformation of ECTM for ρ_1, while Fig. 7 shows the transformation of ECTM for both ρ_1 and ρ_2.

When applying the $ECTM_{SAF}$ transformation, we get:

- (**Rule 1 + Rule 2**) Routers have been removed and the physical links are transformed, in the analysis model, into processing elements.

- (**Rule 3**) All tasks of the architectural model are kept in the analysis model.

- (**Rule 4 + Rule 5**) ρ_1 and ρ_2 use 4 physical links. Thus, the flow ρ_1 has been transformed in the analysis model to a task set Γ_{ρ_1} of 4 tasks: $\Gamma_{\rho_1} = \{ \tau_{\rho_{1,1}}, \tau_{\rho_{1,2}}, \ldots, \tau_{\rho_{1,4}} \}$.

  The flow ρ_2 has been transformed in the analysis model to a task set Γ_{ρ_2} of 4 tasks: $\Gamma_{\rho_2} = \{ \tau_{\rho_{2,1}}, \tau_{\rho_{2,2}}, \ldots, \tau_{\rho_{2,4}} \}$.

  $\tau_{\rho_{1,1}}, \tau_{\rho_{1,2}}, \tau_{\rho_{1,3}}$ and $\tau_{\rho_{1,4}}$ are respectively executed by the processing elements PE_{PE1R1}, PE_{R1R2}, PE_{R2R3} and PE_{R3PE3}.

  $\tau_{\rho_{2,1}}, \tau_{\rho_{2,2}}, \tau_{\rho_{2,3}}$ and $\tau_{\rho_{2,4}}$ are respectively executed by the processing elements PE_{PE2R2}, PE_{R2R3}, PE_{R3R4} and PE_{R4PE4}.

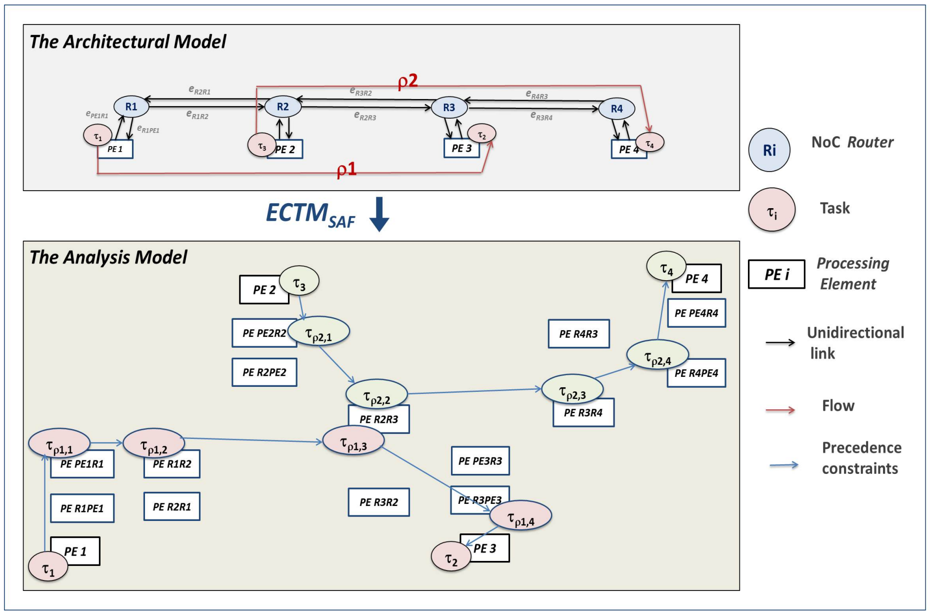

Figure 7: Example of $ECTM_{SAF}$ for two flows

We consider a list scheduling algorithm for processing elements PE_1, PE_2, PE_3 and PE_4. For the rest of the processing elements, a scheduling algorithm in agreement with the considered arbitration policy in the considered NoC is assumed.

In the analysis model, flows ρ_1 and ρ_2 share the same physical link e_{R2R3}. ECTM translates this shared resource in the analysis model by sharing the processing element PE_{R2R3} between $\tau_{\rho_{1,3}}$ and $\tau_{\rho_{2,2}}$.

5.4. Exact Communication Time Model for Wormhole NoC

For the Wormhole NoCs, we propose $ECTM_{Wormhole}$. In the following, we detail rules of $ECTM_{Wormhole}$:

- **Rule 1:** Each router of the architectural model will be removed in the analysis model while keeping all the processing elements.
- **Rule 2:** Each unidirectional link e_{RxRy} in the network between two routers of the architectural model will be modeled in the analysis model by a new processing element PE_{RxRy}.

  Each unidirectional link e_{PExRx} in the network between a processing element and a router (respectively e_{RxPEx} between a router and a processing element) of the architectural model will be replaced in the analysis model by a new processing element PE_{PExRx} (respectively PE_{RxPEx}).

  These new processing elements use a scheduling algorithm in agreement with the considered arbitration policy in the NoC router.

- **Rule 3:** We keep all tasks of the architectural model in the analysis model.
- **Rule 4:** Each flow ρ_i of the architectural model will be replaced by a set of nb tasks Γ_{ρ_i}, where $nb = nblink_i \times nbsize_i$ in the analysis model. $nblink_i$ represents the number of links used by the flow, and $nbsize_i$ represents the size of message of the flow ρ_i.

  $\Gamma_{\rho_i} = \{\ \tau_{\rho_{i,1,1}}, \tau_{\rho_{i,a,b}}, \ \dots\ , \tau_{\rho_{i,nbsize_i,nblink_i}}\}$.

  For example, with $ECTM_{Wormhole}$, a flow of 2 flits, which uses 3 physical links, will be transformed in the analysis model into a task set of 6 tasks.
- **Rule 5:** If the architectural model contains two periodic tasks τ_{source} and $\tau_{destination}$, τ_{source} and $\tau_{destination}$ are executed respectively on the two processing elements PE_s and PE_d. If τ_{source} sends the messages of flow ρ_i to $\tau_{destination}$, then applying $ECTM_{Wormhole}$ leads to the flow ρ_i transformed in the analysis model as a task set Γ_{ρ_i}. Parameters of each task $\tau_{\rho_{i,a,b}}$ of the task set Γ_{ρ_i} are computed as follow:

  For (a,b) $\in$ [1, $size_i$] x [1,$nbrlink_i$], $\tau_{\rho_{i,a,b}}$ is characterized by :

  - $O_{\tau_{\rho_i,a,b}} = O_{\rho_i}$
  - $T_{\tau_{\rho_i,a,b}} = T_{\rho_i}$
  - $C_{\tau_{\rho_i,a,b}} = PD_{Oneflit/Onelink}$

    $PD_{Oneflit/Onelink}$ is the time required to send one flit over only one link without considering the possible conflicts in the network.
  - $D_{\tau_{\rho_i,a,b}} = D_{\rho_i}$
  - $Node_{\tau_{\rho_i,a,b}} = \begin{cases} PE_{PE_sR_s} & if\ a = 1 \\ PE_{R_xR_y} & if\ 1 < a < nbrlink_i \\ PE_{R_dPE_d} & if\ a = nbrlink_i \end{cases}$

    We note here that e_{RxRy} represents one of the physical links used by the flow ρ_i which is transformed to the processing element $PE_{R_xR_y}$ in the analysis model.
  - $E(\tau_{\rho_i,a,b}) = \begin{cases} \tau_{\rho_i,a+1,b}, \tau_{\rho_i,a,b+1} & if\ a < size_i\ and\ b < nbrlink_i \\ \tau_{\rho_i,a+1,b} & if\ a < size_i\ and\ b = nbrlink_i \\ \tau_{\rho_i,a,b+1} & if\ a = size_i\ and\ b < nbrlink_i \\ \tau_{destination} & if\ a = size_i\ and\ b = nbrlink_i \end{cases}$

    We note here that some tasks of Γ_{ρ_i} have only one successor such as $\tau_{\rho_i,size_i,b}$, $\tau_{\rho_i,a,nbrlink_i}$ and $\tau_{\rho_i,size_i,nbrlink_i}$. The rest of the tasks have two successors.

In order to illustrate the transformation of the proposed model for Wormhole NoCs, we apply $ECTM_{Wormhole}$ on the previous example (Fig. 5). Figures 8 and 9 show the architectural and the analysis models with $ECTM_{Wormhole}$. Fig. 8 shows the transformation of ECTM for ρ_1, while Fig. 9 shows the transformation of ECTM for both ρ_1 and ρ_2.

When applying the $ECTM_{Wormhole}$ transformation, we get:

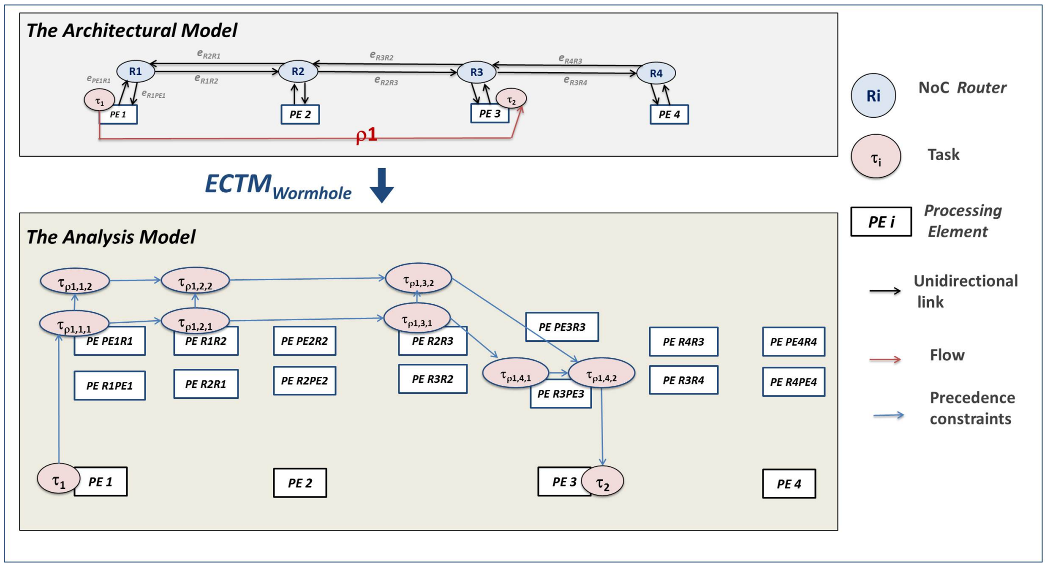

Figure 8: Exact Communication Time Model for Wormhole: Example - 1 flow

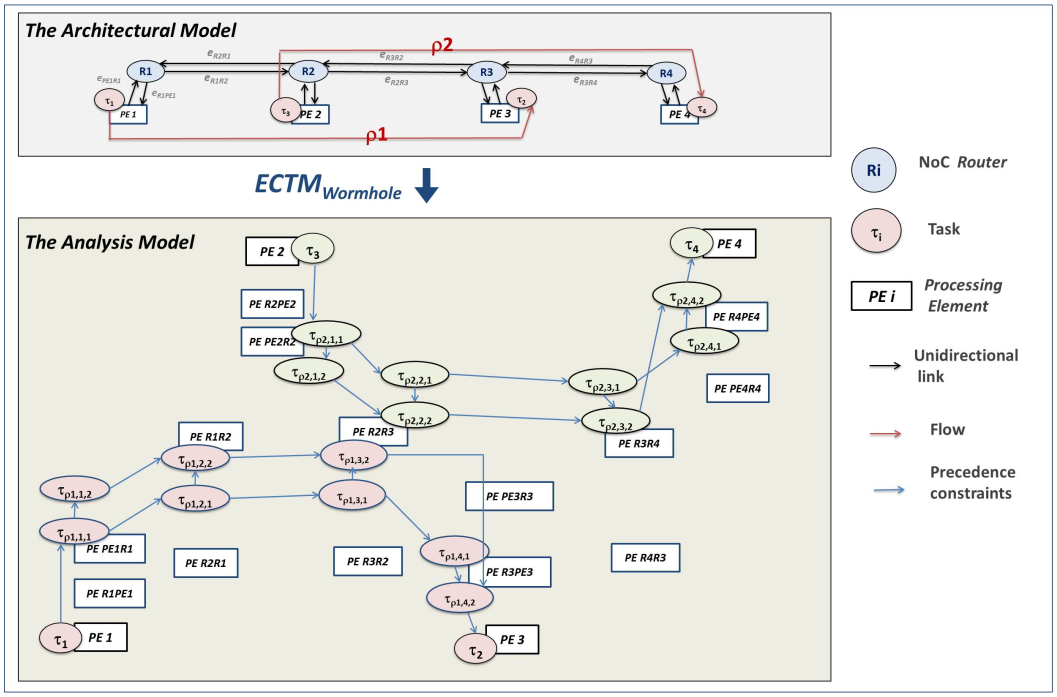

Figure 9: Exact Communication Time Model for Wormhole: Example - 2 flows

- (**Rule 1 + Rule 2**) Routers of the architectural model are removed in the analysis model. Physical links are transformed in the analysis model into processing elements.
- (**Rule 3**) All tasks of the architectural model are kept in the analysis model.
- (**Rule 4 + Rule 5**) The flow ρ_1 and ρ_2 use 4 physical links. The flow size is 2 flits. Thus, the flows ρ_1 and ρ_2 are transformed in the analysis model into 2 task sets Γ_{ρ_1} and Γ_{ρ_2} of 8 tasks:

  $\Gamma_{\rho_1} = \{ \tau_{\rho_{1,1,1}}, \tau_{\rho_{1,1,2}}, \cdots, ..., \tau_{\rho_{1,4,2}}\}$ and $\Gamma_{\rho_2} = \{ \tau_{\rho_{2,1,1}}, \tau_{\rho_{2,1,2}}, \cdots, ..., \tau_{\rho_{2,4,2}}\}$.

In the architectural model, flows ρ_1 and ρ_2 share the same physical link e_{R2R3}. $ECTM_{Wormhole}$ translates this shared resource in the analysis model by sharing the processing element PE_{R2R3} between $\tau_{\rho_{1,3,1}}$, $\tau_{\rho_{1,3,2}}$, $\tau_{\rho_{2,2,1}}$ and $\tau_{\rho_{2,2,2}}$.

After computing the analysis model using ECTM, we perform the scheduling analysis with list scheduling algorithms. In this work, we consider the algorithm *Highest Level First with Estimated Time (HLFET).*

With HLFET, first we assign a priority to each task. Then, a scheduling list of tasks is built. A ready task will be executed over the corresponding processor if available. If two ready tasks share the same corresponding processor, the higher priority task will be executed first. We note that there is no preemption [22, 6].

In the architectural model, at the Wormhole network level, we have flit level preemption. However when applying ECTM, each flit per physical link will be transformed into one task. This is why we use non preemptive scheduling algorithm for the analysis model.

At the SAF network level, we have packet level preemption. When applying ECTM, each packet per physical link will be replaced by one task in the analysis model. Then again, we use non preemptive scheduling algorithm for the analysis model in order to model the fixed priority packet level arbitration of the architectural model.

In the next section, we present a validation of the proposed approach.

6. Validation

For the Wormhole and SAF NoC, ECTM transforms every flow ρ_i of the architectural model to a set of dependent tasks in the analysis model. We show, in this section, that the communication interference caused by hardware shared resources as expressed the architectural model are kept in the analysis model computed by ECTM, leading to a correct schedulability analysis. We express this property below:

Property 1. *Let assume a SAF or Wormhole NoC implementing a fixed priority arbitration policy. Let a task τ_1 that sends a flow ρ_i to the task τ_2. If*

ECTM transforms the flow ρ_i to a set of dependent tasks Γ_{ρ_i}, then the response time of the set Γ_{ρ_i} is equal to the effective communication time of the flow ρ_i.

The following inequality follows from the property 1:

$$PD_{\rho_i} \leq C_{eff_{\rho_i}} = C_{ECTM_{\rho_i}} \leq C_{\rho_i} \tag{1}$$

where

- PD_{ρ_i} is the communication time of the flow ρ_i in absence of communication interference in the NoC.
- $C_{eff_{\rho_i}}$ specifies the effective communication time of the flow ρ_i.
- $C_{ECTM_{\rho_i}}$ is the sum of the capacity of all the tasks of the task set Γ_{ρ_i}.
- C_{ρ_i} specifies the worst case communication time of all messages of the flow ρ_i.

Next, we demonstrate that the property 1 is true for the SAF and Wormhole NoCs. To do so, we consider two cases: with and without interference.

6.1. ECTM for SAF NoC

We consider two dependent periodic tasks τ_1 and τ_2. τ_1 sends the flow ρ_i to the task τ_2. ρ_i uses n physical links. With $ECTM_{SAF}$, the flow ρ_i of the architecture model is transformed into a set of n tasks Γ_{ρ_i} in the analysis model ($\Gamma_{\rho_i} = \{\tau_{\rho_i,1}, ...\tau_{\rho_i,n}\}$)

We remind that under this configuration, we cannot have indirect interference. In addition, the preemption delay is stated as null.

6.1.1. Case without interference

Before starting our demonstration, we remind the path delay equation of the flow ρ_i for such a NoC configuration [3]:

$$PD_{\rho_i} = (size_{max}/B_{link} + R_{hop}) \cdot n = PD_{OneLink} \cdot n \tag{2}$$

where

- B_{link} specifies the link bandwidth for a physical link in the considered NoC.
- $size_{max}$ is the maximum packet size belonging to ρ_i.
- n denotes the number of hops between the source and destination nodes for ρ_i.
- $Flit_{size}$ specifies the flit size.
- R_{hop} indicates the constant processing delay in each router.

In the *without interference* case, we have:

$$
\begin{aligned}
C_{ECTM_{\rho_i}} &= C_{\Gamma_{\rho_i}} && \text{(by definition of } C_{ECTM_{\rho_i}}\text{)} \\
C_{\Gamma_{\rho_i}} &= \sum_{j=0}^{n} C_{\tau_{\rho_i,j}} && \text{(tasks are sequentially dependents)} \\
C_{\Gamma_{\rho_i}} &= \sum_{j=0}^{n} C_{\tau_{\rho_i,1}} && (C_{\tau_{\rho_i,1}} = C_{\tau_{\rho_i,2}} = \cdots = C_{\tau_{\rho_i,n}}) \\
C_{\Gamma_{\rho_i}} &= \sum_{j=0}^{n} PD_{OneLink} && \text{(rule 4 of ECTM i.e. } C(\tau_{\rho_i,1}) = PD_{OneLink}\text{)} \\
C_{\Gamma_{\rho_i}} &= n \cdot PD_{OneLink} \\
C_{\Gamma_{\rho_i}} &= PD_{\rho_1} && \text{(see Eq. 2)}
\end{aligned}
$$

Then, without interference, we know that:

$$PD_{\rho_i} = C_{eff_{\rho_i}} = C_{\rho_i}$$

and then the property 1 est true in that case, and Eq. 1 holds.

6.1.2. *Case with interference*

In this paragraph, we assume that the flow ρ_i shares physical links with j other higher priority flows but only k (with k $\leq$ j) of them actually interfere with it.

We remind equations of the worst communication time and effective communication time of the flow ρ_i for such NoC configurations [3]:

$$C_{\rho_i} = PD_{\rho_i} + \sum_{1}^{j} PD_{OneLink} \tag{3}$$

$$C_{eff_{\rho_i}} = PD_{\rho_i} + \sum_{1}^{k} PD_{OneLink} \tag{4}$$

In this case, we have

$$
\begin{aligned}
C_{ECTM_{\rho_i}} &= C_{\Gamma_{\rho_i}} \\
C_{\Gamma_{\rho_i}} &= PD_{\rho_1} + \sum_{x=1}^{k} C_{\tau_x} && \text{(Rule 3 of ECTM)}
\end{aligned}
$$

Notice that the processing element which executes τ_x tasks, is an abstraction of a NoC router. It applies a scheduling algorithm in agreement with the arbitration policy of the router to ensure that the execution interval of each task of the set Γ_{ρ_i} is equal to the transmission interval of the corresponding message in the architectural model.

Considering Rule 4 of $ECTM_{SAF}$, we have:

$$\begin{aligned}
C_{\Gamma_{\rho_i}} &= PD_{\rho_1} + \sum_{x=1}^{k} PD_{OneLink} && \text{(Rule 4 of ECTM i.e.} \\
& && C(\tau_{\rho_i,1}) = PD_{OneLink}) \\
C_{\Gamma_{\rho_i}} &= PD_{\rho_1} + k \cdot PD_{OneLink} \leq PD_{\rho_1} + j \cdot PD_{OneLink} && (k \leq j) \\
C_{\Gamma_{\rho_i}} &= C_{eff_{\rho_i}} \leq C_{\rho_i} && \text{(Eq. 4 and Eq. 3)}
\end{aligned}$$

So, we can deduce:

$$PD_{\rho_i} \leq C_{eff_{\rho_i}} = C_{ECTM_{\rho_i}} \leq C_{\rho_i}$$

To conclude, ECTM keeps all possible communication interference of the architectural model in the analysis model for a SAF NoC and computes a communication time exactly equal to the effective communication time of the flow ρ_i.

6.2. ECTM for Wormhole NoC

In this paragraph, we consider two dependent periodic tasks τ_1 and τ_2. The task τ_1 sends the flow ρ_i to the task τ_2. The flow goes through m physical links. The size of ρ_i is n flits.

Applying $ECTM_{Wormhole}$, the flow ρ_i of the architectural model will be transformed into a set of $(n \cdot m)$ tasks in the analysis model, leading to the task set $\Gamma_{\rho_i} = \{\tau_{\rho_i,1,1}, ...\tau_{\rho_i,n,m}\}$.

We note that under such NoC configurations (see Tab. 1), we cannot have indirect interference. Indeed, using virtual channel for each flow allows us to avoid indirect interference [18]. Consequently, we have two different cases: without and with direct interference. Next, we verify the property 1 for these two cases.

6.2.1. Case without interference

Before starting our demonstration for Wormhole NoC, we remind the equations of path delay [3] of the flow ρ_i:

$$PD_{\rho_i} = n \cdot PD_{Oneflit/Onelink} + R_{hop} \cdot (m-1) \quad (5)$$

where

- $PD_{Oneflit/Onelink}$ specifies the communication time of one flit over one link without considering the possible conflicts in the network.

- n is the message size of the flow ρ_i.
- m denotes the number of physical links used by the flow ρ_i.

In the *without interference* case, and according to Fig. 10, we have:

$$\begin{aligned}
C_{ECTM_{\rho_i}} &= C_{\Gamma_{\rho_i}} && \text{(by definition of } C_{ECTM_{\rho_i}}\text{)}\\
C_{\Gamma_{\rho_i}} &= n \cdot C_{\tau_{\rho_i,1,1}} + (m-1) \cdot C_{\tau_{\rho_i,1,1}} && \text{(see Fig. 10)}\\
C_{\Gamma_{\rho_i}} &= n \cdot C_{\tau_{\rho_i,1,1}} + (m-1) \cdot Rhop && (Rhop = PD_{Oneflit/Onelink})\\
C_{\Gamma_{\rho_i}} &= n \cdot PD_{Oneflit/Onelink} + (m-1) \cdot Rhop && (C_{\tau_{\rho_i,1,1}} = PD_{Oneflit/Onelink})\\
C_{\Gamma_{\rho_i}} &= PD_{\rho_i} && \text{(see Eq. 5)}
\end{aligned}$$

In addition, without any interference, we know that $PD_{\rho_i} = C_{eff_{\rho_i}} = C_{\rho_i}$. So, we can deduce:

$$PD_{\rho_i} \leq C_{eff_{\rho_i}} = C_{ECTM_{\rho_i}} \leq C_{\rho_i}$$

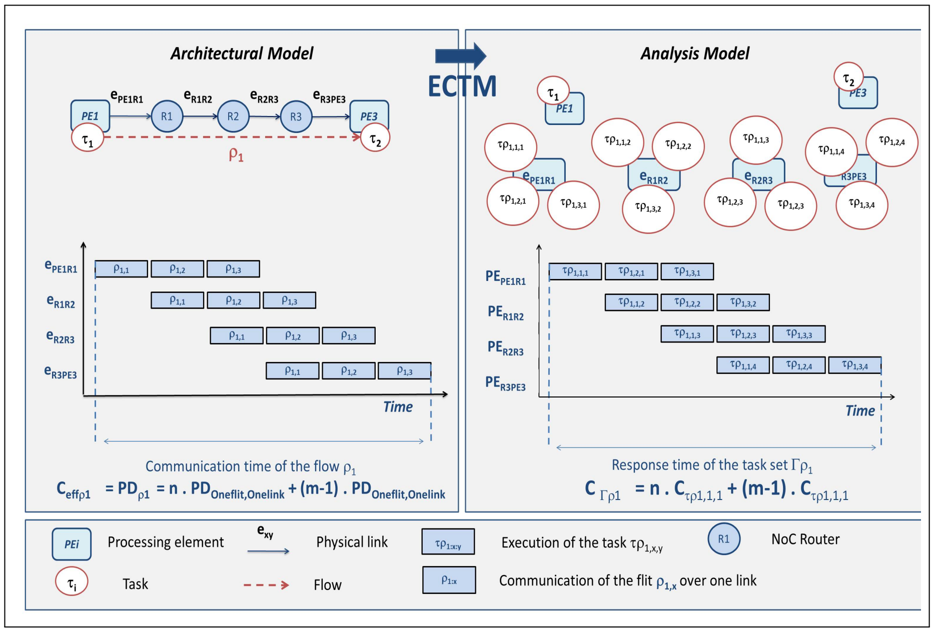
Architectural Model
ECTM
Analysis Model
Communication time of the flow ρ1
Ceffρ1 = PDρ1 = n . PDOneflit,Onelink + (m-1) . PDOneflit,Onelink
Response time of the task set Γρ1
C Γρ1 = n . Cτρ1,1,1 + (m-1) . Cτρ1,1,1
Time
Processing element
Physical link
Execution of the task τρ1,x,y
NoC Router
Task
Flow
Communication of the flit ρ1,x over one link

Figure 10: ECTM transformation for Wormhole NoC

6.2.2. Case with interference

In this paragraph, we assume that the flow ρ_i shares physical links with j other flows but it actually only interferes with k flows (with $k \leq j$).

We note here that the j other flows will be transformed into j task sets in the analysis model $\{ \Gamma_{\rho_1} \cdots \Gamma_{\rho_j} \}$. These task sets will share same processing element with tasks of Γ_{ρ_i}.

We remind the equations of the worst communication time and effective communication time of the flow ρ_i for such a NoC configuration:

$$C_{\rho_i} = PD_{\rho_i} + \sum_{x=1}^{j} PD_{Oneflit/Onelink} \cdot size_{max_x} \tag{6}$$

$$C_{eff_{\rho_i}} = PD_{\rho_i} + \sum_{x=1}^{k} PD_{Oneflit/Onelink} \cdot size_{max_x} \tag{7}$$

where $size_{max_x}$ denotes the maximum message size of the flow ρ_x.

In the *with interference* case, we have:

$$C_{ECTM_{\rho_i}} = C_{\Gamma_{\rho_i}}$$

(by definition of $C_{ECTM_{\rho_i}}$)

$$C_{\Gamma_{\rho_i}} = PD_{\rho_1} + \sum_{y=1}^{k} \sum_{x=1}^{size_{max_x}} C_{\tau_{\rho_i,y,x}}$$

(Rule 3 of ECTM)

$$C_{\Gamma_{\rho_i}} = PD_{\rho_1} + \sum_{y=1}^{k} C_{\tau_{\rho_i,y,1}} \cdot size_{max_x}$$

$(C_{\tau_{\rho_i,y,x}} = C_{\tau_{\rho_i,y,1}})$

$$C_{\Gamma_{\rho_i}} = PD_{\rho_1} + k \cdot C_{\tau_{\rho_i,1,1}} \cdot size_{max_x}$$

$(C_{\tau_{\rho_i,y,1}} = C_{\tau_{\rho_i,1,1}})$

$$C_{\Gamma_{\rho_i}} = PD_{\rho_1} + k \cdot PD_{Oneflit/Onelink} \cdot size_{max_x}$$

$(C_{\tau_{\rho_i,1,1}} = PD_{Oneflit/Onelink})$

$$C_{\Gamma_{\rho_i}} = C_{eff_{\rho_i}} \leq PD_{\rho_1} + j \cdot PD_{Oneflit/Onelink} \cdot size_{max_x}$$

(see Eq. 7 ; $k \leq j$)

$$C_{\Gamma_{\rho_i}} = C_{eff_{\rho_i}} \leq C_{\rho_i}$$

(see Eq. 6)

So, we can deduce that:

$$PD_{\rho_i} \leq C_{eff_{\rho_i}} = C_{ECTM_{\rho_i}} \leq C_{\rho_i}$$

To conclude, ECTM keeps the impact of all possible communication interference of the architectural model in the analysis model for a Wormhole NoC. ECTM provides a communication time that is equal to the effective communication time of the flow ρ_i.

The next section gives an overview of the software framework where the transformations we proposed have been implemented.

7. Implementation and Evaluation

We have produced several experiments in order to evaluate the accuracy and the scalability of the proposed models. To perform this evaluation, we have implemented our communication models into a real-time scheduling simulator called Cheddar [24]. Then, with a first experiment, we evaluate the accuracy of ECTM and WCCTM. A second experiment tests the scalability of our approach. Finally, we present a comparative analysis between ECTM and the holistic verification approach.

7.1. Implementation of ECTM and WCCTM

Cheddar is a GPL framework that provides a scheduling simulator, schedulability tests and various features related to the design and the scheduling analysis of real-time systems. To model the system to analyze, Cheddar provides a specific architecture design language, called CheddarADL [24]. CheddarADL allows users to describe both the software and the hardware parts of the system they expect to analyze.

Fig. 11 is an overview of the software architecture of our prototype and a subset of Cheddar framework libraries.

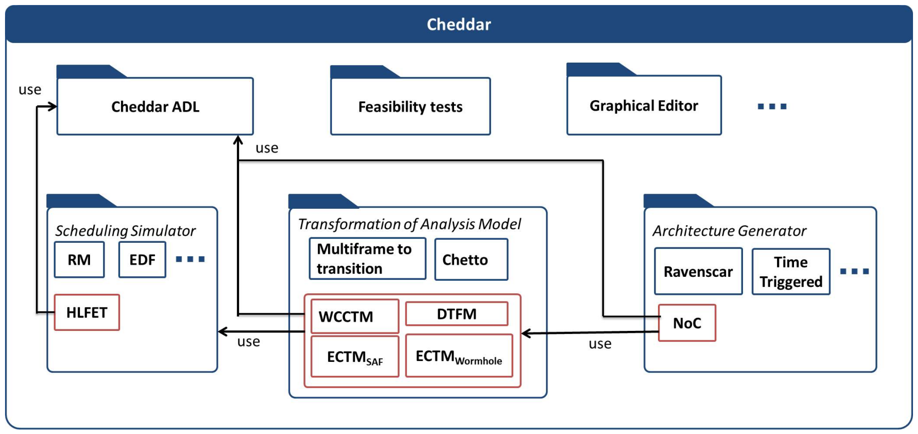

Figure 11: Implementation in Cheddar. The NoC Architecture Generator produces a random real-time system deployed over a NoC architecture. WCCTM and ECTM boxes are for the NoC communication models described in this article. Finally, the module HLFET extends the scheduling algorithms implemented in the scheduling simulator of Cheddar.

7.2. Evaluation process

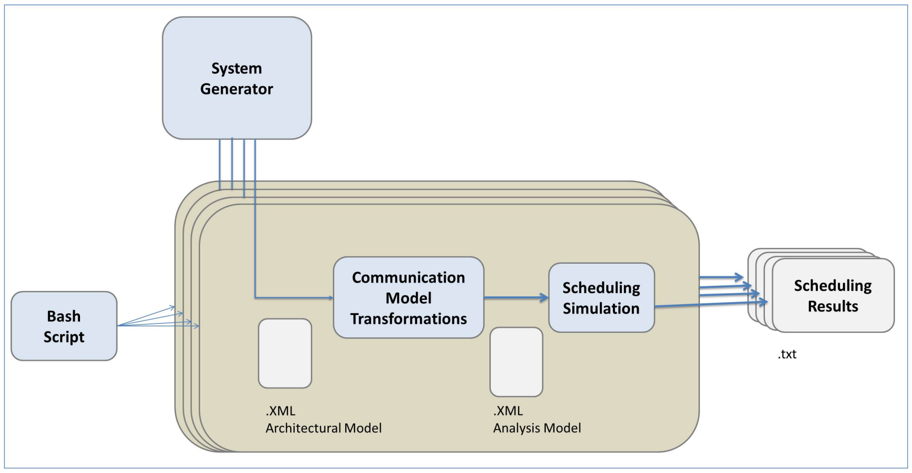

Figure 12: Simulation Details

As shown in Fig. 12, the evaluation process consists of two stages: generation of an architectural model and its scheduling simulation[1].

1. Generation of an architectural model. In this first stage, we produce the input data for the simulations. To do so, we randomly generate a set of architectural models using the UUniFast [25] algorithm. Each architectural model is stored in an XML file which describes the task model, the NoC model, the task mapping and the dependencies between tasks.
2. Scheduling simulation. In this step, we launch a set of simulations. As shown in Fig. 12, for each simulation, we perform the following two operations:
    (a) Model Transformation: The input data of this operation is the architectural model (XML file). Using a model transformation, we generate the analysis model from the architectural model. The model transformations used are WCCTM, $ECTM_{Wormhole}$ and $ECTM_{SAF}$. The analysis models obtained are also stored in XML files.
    (b) Scheduling analysis: We run the scheduling simulation which applies the HLFET list scheduling algorithm. Then, the simulator verifies whether all the deadlines are met.

In the next section, we present the results of the evaluation.

[1]The used artifacts for the evaluation in this work are available from the link `http://beru.univ-brest.fr/svn/CHEDDAR/trunk/artefacts/ae2019/`

7.3. Accuracy of WCCTM and ECTM

The purpose of this evaluation is to make a comparative study between the two ECTM and WCCTM models in terms of accuracy. To do so, we evaluate the rate of schedulable task sets found as schedulable by ECTM and WCCTM models.

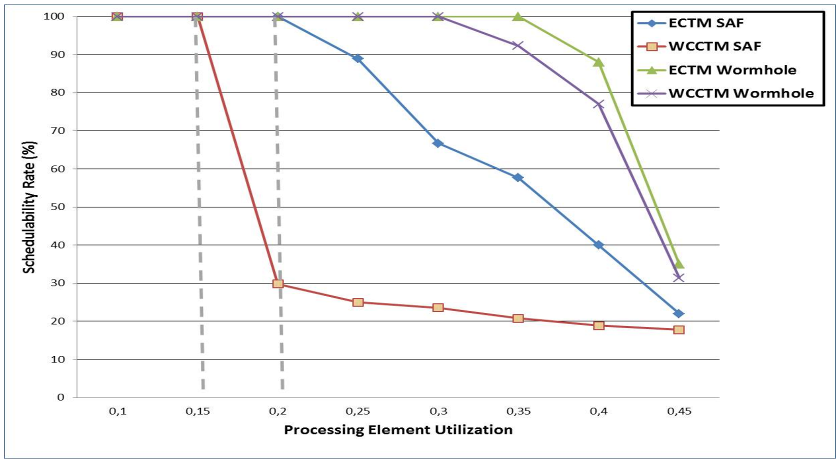

Figure 13: Accuracy of the NoC analysis models for All-To-One traffic

In this experiment, we randomly generate 100 sets of dependent periodic tasks using UUniFast [25]. We choose a random task mapping and we vary the number of flows in the network and the number of tasks.

We consider two traffic patterns: All-To-One and One-To-One.

With a All-To-One traffic pattern, the source node will be randomly selected using UUniFast, while the destination node is fixed arbitrarily. All flows of the same set have the same destination node.

With a One-To-One traffic pattern, the destination node and the source node are selected randomly using UUniFast. The chosen packet size is 4 flits.

To perform the experiment, we first use DTFM in order to compute the flow model from the task model, the task mapping and the NoC model. Then, we apply ECTM or WCCTM and we compute the scheduling simulation with a list scheduling algorithm. Since it presents the best results in term of robustness and complexity, scheduling analysis is performed with HLFET [6]. Simulation intervals are choosen according to [21].

We also consider two NoC models. The first one is a Wormhole NoC, and the second is a SAF NoC. The two NoCs have the same size and the same topology (4×4 2D mesh).

Fig. 13 presents the rate of task sets considered as schedulable according to the overall processing element utilization rate for the All-To-One traffic pattern. The figure shows that ECTM is more accurate than WCCTM with an improvement of 30% for Store-And-Forward NoCs. From a use rate equal to 0.15, some schedulable task sets are no more considered as schedulable by $WCCTM_{SAF}$

model. However, with the $ECTM_{SAF}$, all the task sets remain as schedulable up to a use rate of 0.2.

Fig. 14 presents results equivalent to those shown in Fig. 13 but for the One-To-One traffic pattern. It confirms the previous experimentation.

ECTM is less pessimistic than WCCTM. For a Wormhole NoCs, the threshold of the processing element utilization beyond which some task sets are considered as not schedulable is increased by 100%. Using $WCCTM_{Wormhole}$, the schedulable task sets are no more seen as schedulable from a use rate equal to 0.08. However, with the $ECTM_{Wormhole}$, the schedulable task set is seen as schedulable up to 0.16 of use rate.

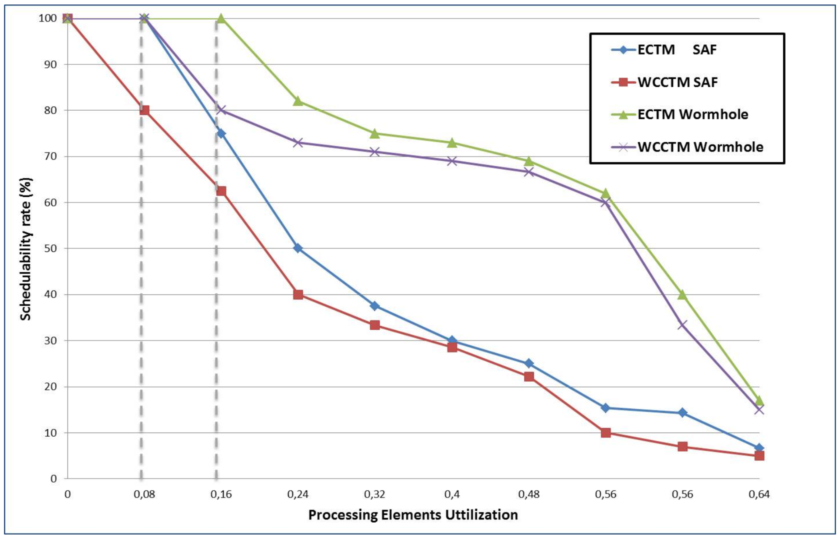

Figure 14: Accuracy of the NoC analysis models for One-To-One uniform traffic

7.4. Evaluation of the scalability of our approach

In order to evaluate the scalability of our approach, we measured the computation time of the WCCTM and ECTM transformations. We keep the same configurations than the previous experiments. Fig. 15 presents this result for different numbers of flows ranging from 15 to 120.

WCCTM provides shorter computation time than ECTM. For 105 flows in the network, WCCTM takes 2.32 seconds to compute the analysis model, while $ECTM_{SAF}$ takes 2.86 seconds, and $ECTM_{Wormhole}$ takes 6.80 seconds for 2 flits flows and 8.13 seconds for 3 flits flows. In terms of computation time, WCCTM has a shorter computation time comparing to ECTM, with a reduction of 17% for the SAF NoCs and upto 54% for the Wormhole NoCs.

7.5. Holistic Analysis versus ECTM

The holistic theory is a formal approach which assesses the feasibility of real-time systems composed of tasks exchanging messages. It makes the analysis of

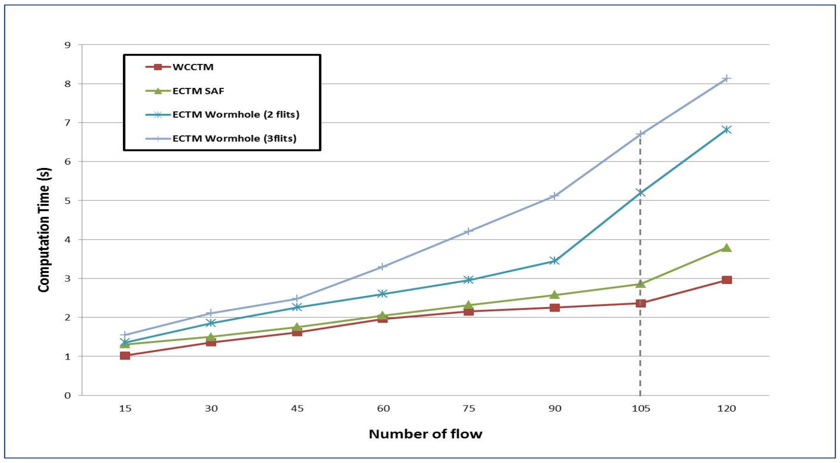

Figure 15: Computation time for the model transformations

distributed systems tractable. In this paragraph, we discuss the efficiency of this approach in the context of NoCs, and then we present a comparative analysis between ECTM and holistic approaches. Finally, we discuss the disavantages and the avantages of each solution.

7.5.1. Holistic approach

The holistic approach has been introduced by Tindell and Clark [26]. The authors describe how to assess the feasibility of real-time distributed systems considering fixed priority tasks and a TDMA (Time Division Multiple Access) network. Later, in [27], Tindell and Burns have extended the analysis to systems with real-time and token communication protocols.

In the holistic approach, feasibility is assessed by computing how timing behavior of the messages affects the response time of the tasks and respectively. In the seminal Tindell proposal, jitter of a task i models lateness introduced by tasks run before task i or messages sending data to task i. The key aspect of the holistic approach is the ability to compute worst case response times for both tasks and messages by a step by step approach. Response times are computed by several approximations upto convergency. At each approximation, for each task i, its jitter is updated by response time approximations of tasks/messages delaying i. Many publications have extended the seminal work of Tindell to various task models, scheduling policie or distributed systems such as [28, 29, 30] and [31].

7.5.2. Holistic NoC communication analysis

Several holistic analyses for distributed systems have been proposed, but in the context of NoCs, the holistic analyses must take into account the specification of the considered switching mode, the arbitration policy and the routing algorithm.

In [7], Shi and Burns have proposed a holistic communication analysis for a 2D mesh Wormhole NoC with XY routing algorithm. In this work, the routers implement a priority based flit-level preemption policy. Many virtual channels are available in the network but each virtual channel is used by only one flow. The authors identify higher priority direct and indirect interference. Then, the path delay of higher flows is affected by lower priority flows which share the same physical link and by flows which suffer higher indirect interference. Similar to the holistic approach proposed by Tindell and Clark, the overall of the different subsystem is repeated several times until stability is reached.

Next, we present a case example in order to compare an holistic approach with ECTM.

7.5.3. A Case example

In the following, we present a comparison, based on an analytical case example, between the ECTM analysis and the holistic analysis proposed by Shi.

Let's take an example, adapted from [7], of a real-time system deployed over a NoC. We consider three communication flows ρ_1, ρ_2 and ρ_3; Table 2 lists the attributes of this flow model. The NoC model, the task model and the task mapping are resumed in Fig. 16. We note that the size of the flow ρ_1 and ρ_3 is 3 flits while the size of ρ_2 is 2 flits. As in the original work presented in [7], we do not consider the task execution time and their impact on the communication time of flows.

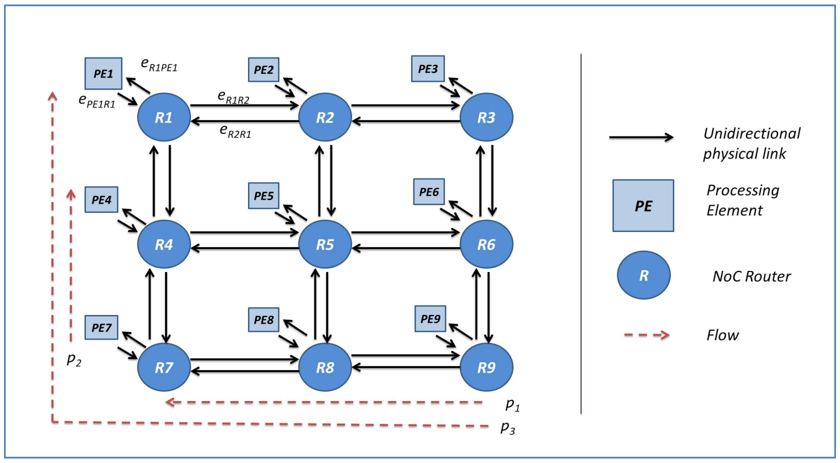

Figure 16: ECTM versus Holistic analysis - case example

Holistic analysis

- ρ_1 and ρ_2 do not suffer any higher direct or indirect interference. Thus, their worst case communication time (R) is equal to their path delay.

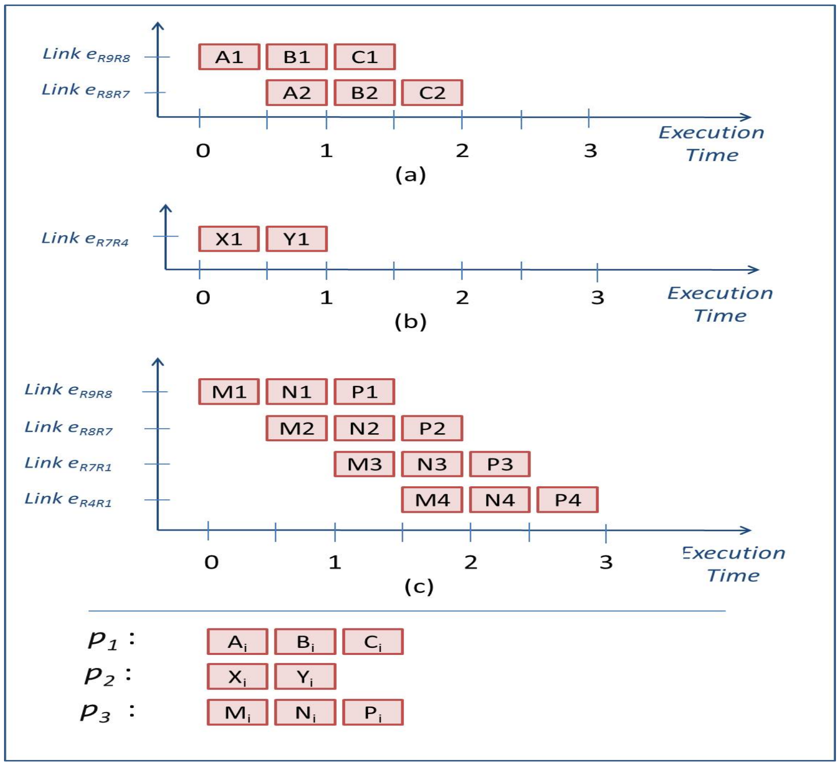

Figure 17: ECTM versus Holistic analysis - case example (a) The path delay for the flow ρ_1. (b) The path delay for the flow ρ_2. (c) The path delay for the flow ρ_3.

Flow	C_i	Π_i	T_i	D_i	J_i^R	Size
ρ_1	2	1	6	6	0	3
ρ_2	1	2	5	5	0	2
ρ_3	3	3	10	10	0	3

Table 2: Case example: Flow model. All the times are expressed as a multiple of a same time unit.

- Shi and Burns have proposed the following equation in [7] in order to compute the worst case communication time of a flow (ρ_i) which suffers higher priority direct interference :

$$R_i = \sum_{j \in S_D} \left\lceil \frac{R_i + J_j^R}{T_j} \right\rceil C_j + C_i$$

  Where

  - R_i represents the worst case communication time of the flow ρ_i.
  - J_j^R represents the jitter of the flow ρ_j.
  - T_j represents the period of the flow ρ_j.
  - C_i is the path delay of the flow ρ_j.
  - C_j denotes the path delay of the flow ρ_j.
  - S_D denotes a set of higher priority flows which share same physical links with ρ_i.

  ρ_3 shares the physical link with the higher priority flows ρ_1 and ρ_2. Thus, the worst case communication time R_3 of the flow ρ_3 is computed and equals to 9 (i.e. $R_3 = 9$).

ECTM

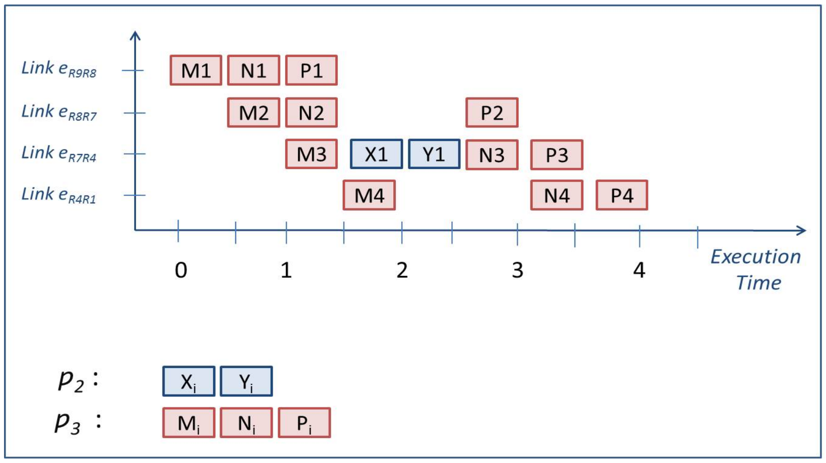

Figure 18: The communication time of ρ_3 computed by ECTM - case example. The flow ρ_3 suffers higher priority direct interference from the flow ρ_1.

ECTM consider the same path delay as the previous analysis. We present the path delay of each flow in Fig. 17

- Similar to holistic analysis, ρ_1 and ρ_2 do not suffer higher priority direct or indirect interference. Thus, their communication time is equal to their path delay.

- ρ_3 may suffer higher direct interference from the flow ρ_1 or/and ρ_2. Considering the emission time of each flow, ECTM can compute which flow may delay ρ_3. In this example, we may have several cases depending on source tasks parameters. In the following, we will specify some of those cases :

  1. No direct interference: In this case, the communication time of ρ_3 is equal to its path delay (see Fig 17 (c)) ($R_3 = 3$).
  2. Direct interference with ρ_1 or ρ_2 for only one time per period: In this case, the flow ρ_3 will suffer higher priority interference from one flow. In Fig. 18 we present the case where the flow ρ_3 suffers a direct interference from the flow ρ_1 for one time during the period of ρ_3. In this case, the communication time for ρ_3 computed by $ECTM$ is (R_3= 4; see Fig. 18).
  3. Direct interference with ρ_1 and ρ_2 for more than one time per period if possible: In this case, the flow ρ_3 will suffer higher priority interference from both ρ_1 and ρ_2 for many times if possible during one period. In this case, we consider the worst case communication time. Thus, ($R_3 = 9$).

In this case example, we do not take into account source and destination tasks in our analysis in order to be consistent with the holistic solution proposed in [7].

As we can see, while holistic analysis only consider the worst case scenario, ECTM is able to consider cases that are under the worst case, leading to a better accuracy of ECTM. However, the disadvantage of the proposed solution is the limitation of its practical exploitation, as for example, we need to extend ECTM to support shared resources or jitter for example. Holistic analysis is less accurate than the proposed solution since it considers worst case scenarios but can be easily extended to support extra sources of delay, as it was extended for jitter or shared resources.

8. Conclusion

Delays introduced by a NoC make the schedulability analysis challenging. Classic NoC real-time scheduling solutions consider worst case scenarios to analyse the communication schedulability instead of the actual delays introduced by the network and lead to pessimistic schedulability analysis results.

In this article, we propose a new NoC communication model called Exact Communication Time Model (ECTM) in order to analyze the scheduling of periodic tasks exchanging messages over a NoC. Our approach supports Wormhole and Store-And-Forward NoC switching techniques. With ECTM, we perform scheduling analysis with a list scheduling algorithm called HLFET.

We have implemented our approach in a real-time scheduling simulator called Cheddar. The results show that ECTM is more efficient than the classic solution WCCTM with an improvement of 30% for Store-And-Forward NoCs and of

100% for Wormhole NoCs, while in terms of computation time of the analysis model, WCCTM is better than ECTM with 17% for Store-And-Forward NoCs and with 54% for wormhole NoCs.

ECTM only captures the interference the flows meet in the network by considering the temporal parameters of the task and flow models contrary to classic solutions based on WCCTM. This explains why ECTM is more accurate for the schedulability analysis of the system.

On the analysis overhead point of view, a larger computation time is required for ECTM which is explained by the complexity of ECTM transformations. The computation time is proportional to the message size and the number of used physical links. Moreover, ECTM does not consider all type of interference such as indirect interference, and then, ECTM is not able to support all configurations of NoCs. ECTM can be applied to virtual channel SAF NoC and private virtual channel wormhole NoC.

In future work, we will evaluate the overhead on the computation time introduced by the proposed model on the scheduling analysis. Furthermore, we intend to use ECTM and its associated tools to investigate the scheduling analysis of mixed criticality systems deployed over NoC architectures. We will also extend ECTM to support *SpaceWire* networks [32]. In this technology, flows are transferred with Wormhole switching mode without using virtual channel. Consequently, we cannot avoid indirect interference situations in *SpaceWire* networks. We investigate a possible solution to master such indirect interference by using shared resources in the computed analysis model.

Acknowledgements

This work and Cheddar[2] are supported by Brest Métropole, Ellidiss Technologies, CR de Bretagne, CD du Finistère, H2020 MOSAR and Campus France PESSOA programs 27380SA and 37932TF.

The authors would like to thank all the supporting staff and partners of the MOSAR Project: Space Applications Services N.V. (Belgium), Deutsches Zentrum Fuer Luft - Und Raumfahrt Ev (DLR), GMV Aerospace And Defence Sa (GMV), Thales Alenia Space UK Ltd (TAS-UK), Sitael Spa (SITAEL), MAG SOAR S.L. (MAG SOAR S.L.), the University of Strathclyde (UK), Thales Alenia Space S.p.A (France) and Ellidiss Technologies (ELLIDISS). MOSAR is part of the PERASPERA project on Space Robotics Technologies, and has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 821996.

References

[1] S. Ma, L. Huang, M. Lai, W. Shi, Networks-on-Chip: From implementation to programming paradigms, Morgan Kaufmann, 2014.

[2] `http://beru.univ-brest.fr/~singhoff/cheddar/`

[2] S.-C. Cheng, J.-A. Stankovic, K. Ramamritham, Tutorial: Hard real-time systems, IEEE Computer Society Press, Los Alamitos, CA, USA, 1989, Ch. Scheduling Algorithms for Hard Real-time Systems: A Brief Survey, pp. 150–173.

[3] Z. Shi, Real-time communication services for networks on chip, Ph.D. thesis, University of York (Nov 2009).

[4] R. Pop, S. Kumar, A survey of techniques for mapping and scheduling applications to network on chip systems, Tech. Rep. 04:4, Department of Electronics and Computer Engineering School of Engineering, Jönköping University (2004).

[5] M. Qamhieh, Scheduling of parallel real-time DAG tasks on multiprocessor systems, Ph.D. thesis, Université Paris-Est (Jan. 2015).

[6] T. L. Adam, K. M. Chandy, J. R. Dickson, A comparison of list schedules for parallel processing systems, Communications of the ACM 17 (12) (1974) 685–690.

[7] Z. Shi, A. Burns, Real time communication analysis for on-chip networks with wormhole switching, in: Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip (NOCS), 2008, pp. 161–170.

[8] Z. Shi, A. Burns, Real-time communication analysis with a priority share policy in on-chip networks, in: Proceedings of the 21^{st} Euromicro Conference on Real-Time Systems (ECRTS), 2009, pp. 3–12.

[9] M. B. Nejad, E. B. Nejad, A. Sayahi, S. M. Hashemi, J. Chaharlang, Mapping and scheduling techniques for network-on-chip architecture, International Journal of Basic Sciences and Applied Research 2 (7) (2013) 686–693.

[10] G. Varatkar, R. Marculescu, Communication-aware task scheduling and voltage selection for total systems energy minimization, in: Proceedings of the 2003 International Conference on Computer Aided Design (ICCAD), 2003, pp. 510–517.

[11] T. Lei, S. Kumar, A two-step genetic algorithm for mapping task graphs to a network on chip architecture, in: Proceedings of the Euromicro Symposium on Digital System Design, 2003, pp. 180–187.

[12] T. Lei, S. Kumar, Algorithms and tools for network on chip based system design, in: Proceedings of the 16^{th} Symposium on Integrated Circuits and Systems Design, 2003, pp. 163–168.

[13] M. Qamhieh, L. George, S. Midonnet, A Stretching Algorithm for Parallel Real-time DAG Tasks on Multiprocessor Systems, in: Proceedings of the 22^{nd} International Conference on Real-Time Networks and Systems, Versaille, France, 2014, pp. 13–22.

[14] K. Lakshmanan, S. Kato, R. Rajkumar, Scheduling parallel real-time tasks on multi-core processors, in: proceedings of the 31^{st} IEEE Real-Time Systems Symposium, 2010, pp. 259–268.

[15] K. Jetly, Experimental comparison of store-and-forward and wormhole NoC routers for FPGAs, Ph.D. thesis, University of Windsor (Nov 2013).

[16] M. Dridi, S. Rubini, F. Singhoff, J.-P. Diguet, DTFM: a flexible model for schedulability analysis of real-time applications on noc-based architectures, in: Proceeding of the 4^{th} IEEE International Workshop on Real-Time Computing and Distributed systems in Emerging Applications (REACTION), 2016, pp. 43–49.

[17] P. H. Feiler, D. P. Gluch, Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language, Addison-Wesley, 2012.

[18] M. Dridi, S. Rubini, M. Lallali, M. J. S. Flórez, F. Singhoff, J.-P. Diguet, Design and multi-abstraction-level evaluation of a noc router for mixed-criticality real-time systems, ACM Journal on Emerging Technologies in Computing Systems 15 (1) (2019) 2:1–2:37.

[19] M. A. Al Faruque, J. Henkel, Minimizing virtual channel buffer for routers in on-chip communication architectures, in: Proceedings of the Design Automation and Test Europe Conference (DATE), 2008, pp. 1238–1243.

[20] M. A. Al Faruque, J. Henkel, Transaction specific virtual channel allocation in QoS supported on-chip communication, in: Proceedings of the IEEE International Conf on Application-specific Systems, Architectures and Processors (ASAP), 2007, pp. 48–53.

[21] J. Goossens, E. Grolleau, L. Cucu-Grosjean, Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms, Real-Time Systems 52 (6) (2016) 808–832.

[22] T. Hagras, J. Janeček, Static vs. dynamic list-scheduling performance comparison, Acta Polytechnica 43 (6) (2003).

[23] Y.-K. Kwok, I. Ahmad, Static scheduling algorithms for allocating directed task graphs to multiprocessors, ACM Computing Surveys 31 (4) (1999) pp. 406–471.

[24] F. Singhoff, J. Legrand, L. Nana, L. Marcé, Cheddar: a flexible real time scheduling framework, in: ACM SIGAda Ada Letters, Vol. 24, ACM, 2004, pp. 1–8.

[25] E. Bini, G. C. Buttazzo, Measuring the performance of schedulability tests, Real-Time Systems 30 (1-2) (2005) 129–154.

[26] K. Tindell, J. Clark, Holistic schedulability analysis for distributed hard real-time systems, Microprocessing and Microprogramming 40 (2) (1994) 117–134, parallel Processing in Embedded Real-time Systems.

[27] K. Tindell, A. Burns, A. J. Wellings, Analysis of hard real-time communications, Real-Time Systems 9 (2) (1995) 147–171.

[28] J. J. G. Garcia, J. C. P. Gutierrez, M. G. Harbour, Schedulability analysis of distributed hard real-time systems with multiple-event synchronization, in: Proceedings 12^{th} Euromicro Conference on Real-Time Systems (RTS), 2000, pp. 15–24.

[29] J. C. Palencia, M. G. Harbour, Schedulability analysis for tasks with static and dynamic offsets, in: Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No.98CB36279), 1998, pp. 26–37.

[30] K. Tindell, Adding time-offsets to schedulability analysis, in: Real-Time Research Group, Departement of Computer Science, University of York, England, 1994.

[31] J. C. Palencia, M. G. Harbour, Response time analysis of edf distributed real-time systems, Embedded Computing Journal 1 (2) (2005) 225237.

[32] S. M. Parkes, P. Armbruster, SpaceWire: a spacecraft onboard network for real-time communications, in: 14th IEEE-NPSS Real Time Conference, 2005., 2005, pp. 6–10.