CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Technical Report

Borislav Nikolic

Stefan M. Petters

CISTER-TR-141006

12, Oct, 2014

Borislav Nikolic, Stefan M. Petters

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: +351.22.8340509

E-mail: borni@isep.ipp.pt, smp@isep.ipp.pt

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

Abstract

EDF as an Arbitration Policy for Wormhole-Switched Priority-Preemptive NoCs – Myth or Fact?*

Borislav Nikolić and Stefan M. Petters
CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Portugal
{borni, smp}@isep.ipp.pt

ABSTRACT

A constant increase in the number of processors integrated within multiprocessor platforms led to more apparent contentions for the interconnect medium. Consequently, inter-processor communication latencies significantly outgrew the threshold until which their effects on the real-time analysis of multiprocessors can be discarded as negligible. Yet, despite its ever increasing importance, the contention analysis of interconnects is still in its infancy! In that vein, we propose a novel arbitration policy for interconnect routers, which is based on the EDF paradigm – a well-established approach in the scheduling theory. First, we elaborate on the practical aspects of this model and propose the worst-case traffic delay analysis. Then, we experimentally evaluate the approach against the state-of-the-art methods, and also investigate its practical limitations, so as to give a complete answer to the question posed in the title of this work.

Categories and Subject Descriptors

C.3 [**Special-purpose and application-based systems**]: Real-time and embedded systems

Keywords

Real-Time Systems, Embedded Systems, Multiprocessors, NoCs, Wormhole Switching

1. INTRODUCTION

Slowly but steadily, multiprocessors pave their path into the real-time embedded domain. These platforms offer several beneficial possibilities, for instance, to enhance existing functionalities or to integrate new ones, to perform energy and thermal management, to implement fault tolerance mechanisms. However, the real-time analysis of multiprocessors is a challenging topic, and one of the most notable reasons is the contention for the interconnect medium. This problem can be circumvented for platforms that consist of up to a dozen processors, because the effects of the interconnect contentions are mild and can be disregarded from the real-time analysis. Yet, as the number of processors integrated within multiprocessor platforms doubled, and kept increasing even more, contentions for the interconnect medium became more apparent. Consequently, inter-processor communication latencies significantly crossed the threshold until which their effects can be considered negligible. Therefore, the contention analysis of interconnects has to become an integral part of the real-time analysis of multiprocessors.

The Network-on-Chip architecture [2] (NoC) became the prevailing interconnect medium and mainstream for multiprocessors [9, 21], due to its scalability potential [10]. For currently available NoCs, the *wormhole-switching* [13] is the predominant data transfer technique, because of its good throughput and small buffering requirements [10]. When designing wormhole-switched NoCs, chip manufacturers employ a wide range of diverse design choices and strategies, e.g. different topologies, different sizes of basic transferable units – *flits*, different arbitration policies, different router operating frequencies, the (in)existence of virtual channels. These design trade-offs have a significant impact on both the performance and the analysis. Yet, despite its ever increasing importance, the contention analysis of NoCs, so far, did not receive an adequate attention. In that vein, this work focuses on the real-time analysis of NoCs.

Contribution: We propose a novel arbitration policy for NoC routers, which is based on the EDF paradigm [12] – a well-established concept in the scheduling theory. This work presents the first attempt to consider dynamically changing network traffic priorities for wormhole-switched priority-preemptive NoCs. Specifically, we try to answer the following questions: **What are the prerequisites to enforce the EDF arbitration policy within NoC routers?** (Section 5.1) **How to perform the worst-case traffic delay analysis?** (Section 5.2) **Does this approach outperform the state-of-the-art methods, under which conditions and by how much?** (Section 6) **What are the practical limitations of the approach?** (Sections 6-7) **Ultimately, do its merits justify the overhead of enforcing a novel arbitration policy?** (Section 7).

*This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within projects ref. FCOMP-01-0124-FEDER-037281 (CISTER), FCOMP-01-0124-FEDER-020536 (SMARTS), and by FCT and ESF (European Social Fund) through POPH (Portuguese Human Potential Operational Program), under PhD grant SFRH/BD/81087/2011.

ESWEEK'14, October 12 - 17 2014, New Delhi, India

http://dx.doi.org/10.1145/2656045.2656056.

2. RELATED WORK

The wormhole switching technique [13] was proposed a long time ago, however, for many years it has been neglected because the alternative *store-and-forward switching* technique was providing satisfactory results. As the amount of data that has to be transferred kept increasing, the buffering within routers became a challenge, which lately brought the wormhole-switching back into focus. Nowadays, this method is the predominant switching technique for NoCs, e.g. [9, 21].

If a platform provides only a single channel per link [21], complex contention patterns may occur [11]. Several techniques have been proposed to provide an upper-bound on the worst-case delays of individual traffic flows [5, 6, 7, 15], however, due to complex interference patterns, these methods yield pessimistic results [5].

Conversely, if virtual channels [3, 4] are available within the platform [9], the benefits are twofold: (i) the performance (throughput) can be significantly improved [3, 4] (see Section 4.1), and (ii) preemptions among traffic packets can be implemented [19]. The second aspect is of particular interest for the real-time domain. By following that intuition and by employing several additional assumptions, namely (i) per-traffic-flow distinctive priorities, (ii) per-priority virtual channels, and (iii) flit-level preemptions, Shi and Burns [17] proposed the analysis to compute the upper-bound on the worst-case delays of traffic flows. In order to minimise the number of employed virtual channels, the same authors presented a method [18] which allows multiple flows to share the same channel, although, by incurring additional delay. Nikolić et al. [14] proposed a method which allows to significantly reduce the number of employed virtual channels, without any impact on the analysis, that is, the delays of flows are identical to those from scenarios where each flow has a dedicated virtual channel. Moreover, Shi and Burns [16] proposed a heuristic-based priority assignment technique, which finds a priority ordering among traffic flows, if one exists, such that all flows meet their deadlines. All the aforementioned approaches are based on the underlying assumption that each flow has a fixed priority. Yet, no work has considered NoC routers with arbitration policies which allow dynamically changing flow priorities.

3. MODEL

3.1 Platform

A platform under consideration is a multiprocessor system comprised of $m \times n$ tiles, interconnected by a 2-D mesh NoC interconnect. A tile contains a single processor and a single router (Figure 1). A router has a set of ports, which are used to exchange the data with the neighbouring routers. Each pair of communicating ports is connected with two unidirectional links. The platform employs a static, dimension-ordered XY routing policy, which is deadlock and livelock free [8]. With this technique, packets firstly travel along the x-axis, and upon reaching the x-coordinate of the destination, continue along the y-axis. Moreover, the wormhole-switching technique with the credit-based flow control mechanism is used. This means that, prior to sending, each data packet is divided into small elements of fixed size called *flits*. A header flit establishes the path, and the rest follow in a pipeline manner, while credits traverse separate physical links in the opposite direction. Additionally, the platform

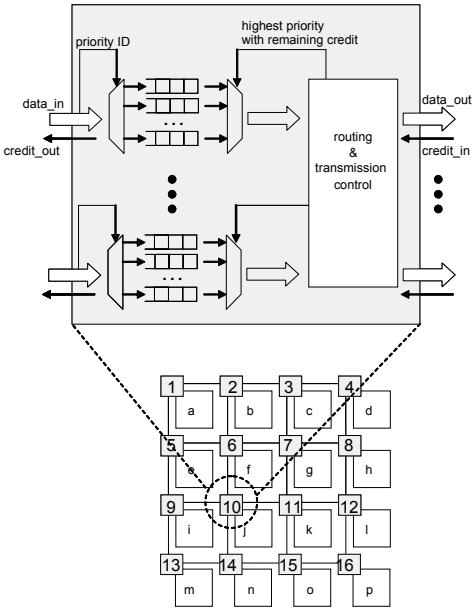

Figure 1: Platform and Router Architecture

provides virtual channels. A virtual channel is an additional buffer within a port of a router, which allows to store flits of preempted packets (see Section 4.1). The number of virtual channels should be at least equal to the maximum number of contentions at any router, which guarantees that each packet will have an available virtual channel within each port along its path [14]. The router architecture is depicted in Figure 1.

3.2 Workload

The inter-processor communication is modelled by a sporadic flow-set $\mathcal{F}$, which is a collection of flows $\{f_1, f_2, ...f_z\}$. Each flow f_i is characterised by a source router $src(f_i)$, a destination router $dst(f_i)$, a set of traversed links $path(f_i)$, a number of hops $hops(f_i)$, its size $size(f_i)$, its minimum inter-arrival period T_i and its deadline D_i. Each flow f_i generates a potentially infinite sequence of packets. A packet released at time instant t should be received no later than $t + D_i$. Otherwise, it has *missed a deadline*. In this work we assume that all flows have implicit deadlines ($\forall f_i \in \mathcal{F} : D_i = T_i$). A reader who is experienced in the scheduling theory may notice that the relationship between flows and packets is identical to that of tasks and jobs. In fact, the NoC contention analysis is very similar to the scheduling theory and we will emphasize other similarities and differences as we encounter them throughout the paper.

4. BACKGROUND AND PRELIMINARIES

In this section we will explain the basic concepts and the state-of-the-art in the NoC contention analysis.

4.1 Significance of Virtual Channels

Traditional wormhole-switched interconnects have only a single channel per direction per link [21]. This implies that, once flits occupy the buffers in the ports along the path of the packet, the other packets which traverse the same path have to wait until the buffers become empty, i.e. until the flits of the existing packet leave the contending buffers. This can cause very complex contention scenarios [11], where a packet can be blocked not only by the packets with which it

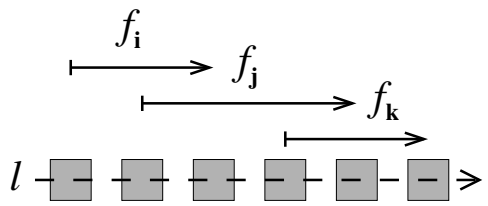

Figure 2: Example of Contending Flows

shares a part of the path, but also by the packets with which it does not. An illustrative example is given in Figure 2, where the flows f_i, f_j and f_k contend for some links on the path l. The shaded rectangles denote routers, while a tail and a head of an arrow depict locations of a source and a destination router of a flow, respectively. It is visible that f_i can be blocked by f_j, which in turn can be blocked by f_k. Thus, f_i can be blocked by f_k even though they *do not* have a common part of the path. Notice, that these effects can cause a significant platform underutilisation.

In order to improve the performance, *virtual channels* have been proposed [3, 4]. Virtual channels give the possibility to store stalled flits of blocked packets, and offer the progress to some other non-blocked packets requesting the same links. In the very same example from Figure 2, once f_j gets blocked by f_k, the flits of its packet can be stored inside the virtual channels, and the packet of f_i can freely progress. The Single-Chip-Cloud Computer [9] employs the wormhole switching with virtual channels. Notice, that virtual channels can also be utilised to enforce flit-level preemptions [19]. That is, once a higher priority flow encounters a progressing lower priority flow, the latter can be stored inside the virtual channels and the progress can be offered to the former.

4.2 Inter-Packet Relationships

Figure 2 suggests that, when a packet of a flow traverses the path from its source to its destination, it can encounter other packets with which it shares some parts of the path. Subsequently, these packets can cause an increase in each other's delay. Thus, the delay of a single packet traversal consists of several components. The first is the isolation delay (Equation 1), or what is in the literature also known as the *basic network latency*. It is interpreted as the delay of the first flit to reach the destination, augmented by the transfer delay of the rest of the flits. d_R and d_L denote the latencies of one flit to traverse one router and one link, respectively, while $size(f)$ represents the size of one flit.

$$C_i = nhops(f_i) \times (d_R + d_L) + \left\lceil \frac{size(f_i)}{size(f)} \right\rceil \times d_L \tag{1}$$

Additionally, due to flit-level preemptions, a packet may be blocked within each router by one lower-priority packet for at most the duration of one flit traversal. The worst-case occurs when the packet experiences this scenario within every router on its path (Equation 2).

$$B_i = nhops(f_i) \times (d_R + d_L) \tag{2}$$

Finally, once a packet encounters a higher-priority packet, it gets preempted. The worst-case interference, caused to the packet under analysis by a packet of the higher-priority flow f_h, is equal to the sum of its isolation and blocking delay – $C_h + B_h$. Let us denote by I_i the maximum joint interference that a packet of the flow under analysis f_i can suffer from other higher-priority packets. Notice, that both B_i and I_i depend on the arbitration policy. By assuming that a packet suffers lower-priority blocking within every router on its path (Equation 2), the term B_i becomes independent of the arbitration policy. However, similar approach is not applicable to the interference component. So, at this stage, let us temporarily assume that I_i has been obtained, and we will explain that process later when we introduce the arbitration policies of interest.

After identifying all these components, now we can define the *worst-case traversal time* – R_i, which is, for a particular flow f_i, equal to the maximum delay that any of its packets may experience (Equation 3).

$$R_i = C_i + B_i + I_i \tag{3}$$

If $R_i \leq D_i = T_i$, then the flow is *schedulable*. The entire flow-set is schedulable, if all its flows are schedulable. Notice the similarities between the worst-case traversal time and the *worst-case response time* from the scheduling theory. Also notice, that in the NoC contention analysis, a path that a packet traverses is considered as a single indivisible resource, and any request for any of its parts by higher-priority packets is treated as interference. A corresponding equivalent in the scheduling theory is a job which is executed on the uniprocessor platform and which suffers the preemptions by higher-priority jobs.

In spite of these similarities, there are some differences as well. For instance, the *transitivity property* may not hold in the NoC context. We show that with the illustrative example given in Figure 2. Assume that the flow f_i can preempt the flow f_j, which in turn can preempt the flow f_k. However, this does not mean that f_i can also preempt f_k. Yet, it is trivial to see that in the uniprocessor scheduling theory the transitivity property applied to job preemptions holds. This difference has several interesting implications which defy the straightforward application of the techniques from the scheduling theory into the NoC contention analysis.

4.3 Fixed-Priority Arbitration Policy

Let us compute the worst-case traversal times for flows from Figure 2, assuming flow characteristics from Table 1.

Table 1: Example of flow characteristics

Flow	Priority	C	B	D = T
f_i	P_i	3	0	10
f_j	$P_k < P_i$	2	0	6
f_k	$P_j < P_k$	2	0	5

If we straightforwardly apply the concepts from the uniprocessor scheduling theory, the total interference that the flow under analysis f_i suffers is computed by summing up the interferences that all higher-priority contending flows $hp(f_i)$ can cause. Individual terms are obtained by multiplying the maximum number of occurrences of a higher-priority flow within the time interval t, with the maximum interference its one packet can cause (Equation 4).

$$I_i = \sum_{\forall f_h \in hp(f_i)} \left\lceil \frac{t}{T_h} \right\rceil \times (C_h + B_h) \tag{4}$$

If we apply this approach to the given example, the computation renders the following results:

$$R_i = C_i + B_i = 3$$

$$R_j = C_j + B_j + \left\lceil \frac{R_j}{T_i} \right\rceil \times (C_i + B_i) = 5$$

$$R_k = C_k + B_k + \left\lceil \frac{R_k}{T_j} \right\rceil \times (C_j + B_j) = 4$$

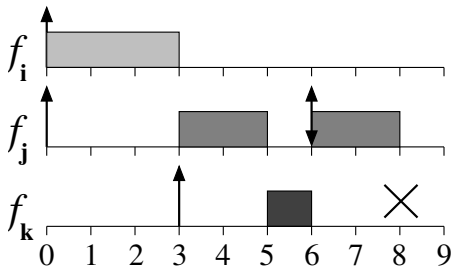

Figure 3: Deadline Miss Example

Since the worst-case traversal times of all the flows are less than their respective deadlines, we may conclude that this flow-set is schedulable. **But that is not true!** Figure 3 demonstrates that the packet of f_k can miss its deadline. The explanation is as follows. Even though f_i cannot directly interfere with f_k because they do not have a common part of the path, f_i can influence the occurrence pattern of f_j and in that way indirectly contribute to the delay of f_k. Notice in Figure 3 that f_i delayed the first packet of f_j, causing its two successive packets to be distanced by less than T_j. Consequently, f_k experienced more interference from f_j than what was computed with Equation 4. In particular, within the observed interval, f_k suffered the interference from two packets of f_j, while the analysis considered the interference from only one packet. Thus, assuming periodic occurrences of higher-priority flows is not safe.

The ability of a flow to influence the delay of another flow with which it does not have a common part of the path is in the literature referred to as the *indirect interference*. Assuming that flow priorities remain constant (the fixed-priority model), Shi and Burns [17] proposed the analysis which takes into account the effects of indirect interferences. The intuition behind their approach is the following: for each higher-priority contending flow which releases can be deferred due to indirect interferences, assume that the first packet is delayed as much as possible, while all the other packets are released as early as possible. The maximum delay that the first packet may experience while still being schedulable is $J_i = R_i - C_i$, which is in the literature known as the *maximum network jitter*. By assuming the maximum jitter for each higher-priority contending flow f_h that is involved in at least one indirect interference relationship (from the perspective of the analysed flow f_i), a safe upper-bound on the interference that f_i suffers within the time interval t is computed by solving Equation 5. Note, if f_h is *not* involved in any indirect interference relationship (from the perspective of f_i), then its jitter is equal to zero.

$$I_i = \sum_{\forall f_h \in hp(f_i)} \left\lceil \frac{t + J_h}{T_h} \right\rceil \times (C_h + B_h) \quad (5)$$

Let us recompute the worst-case traversal time of the flow f_k. The indirect interference that f_i causes to f_k is manifested with the jitter of f_j, i.e. $J_j = R_j - C_j = 3$. And indeed we can see that f_k is unschedulable:

$$R_k = C_k + B_k + \left\lceil \frac{R_k + R_j - C_j}{T_j} \right\rceil \times (C_j + B_j) = 6 > D_k$$

It comes as no surprise that models which allow such deferrals in occurrence patterns are in the scheduling theory called *models with jitters*. However, jitters in the scheduling theory are usually considered as inputs, constants, already specified values, while in the NoC contention analysis, as we noticed, jitters are caused by indirect interferences and highly depend on flow-set characteristics (e.g. flow paths, flow sizes, priorities) and above all – on the arbitration policy! This brings us to the first major difference between the NoC contention analysis and the uniprocessor scheduling theory. For the latter it is well-known that the rate-monotonic is the optimal priority assignment policy in the sense that if a priority assignment exists with which a task-set is schedule, then the task-set is also schedulable with the priorities assigned according to the rate-monotonic policy [12]. However, the fact that the transitivity property does not hold, and the existence of indirect interferences render the same conclusion invalid in the NoC context [16].

5. EDF AS AN ARBITRATION POLICY

5.1 Prerequisites

EDF (Earliest Deadline First) is a well-known concept in the uniprocessor scheduling theory [12]. EDF has been proven optimal in the following sense: if any scheduling policy (including the previously mentioned fixed-priority ones) can render the task-set schedulable, EDF will also be able to do so. EDF has also been studied from the perspective of multiprocessors [1], however, it faces several challenges in that context: (i) it is shown that it is not very efficient, and (ii) due to the necessity to maintain global structures e.g. a ready-queue, scalability issues may arise. So far, no work has considered EDF as an arbitration policy for NoCs and this work is motivated by that fact.

Irrespective of whether it is applied to the scheduling or the NoC contention theory, the EDF policy arbitrates the access to the shared resource (e.g. a processor, a link) based on the latest time instant until which contending entities (e.g. jobs, packets) have to complete their execution. Thus, one of the prerequisites to enforce such a policy in NoC routers is that, prior to sending, at time instant t, a packet of a flow f_i is tagged with its deadline, expressed in absolute values $d_i = t + D_i$. In the ideal case, if two packets, belonging to two distinctive flows f_i and f_j, with their respective deadlines d_i and d_j, encounter each other and contend for some link on their paths, the one with the earlier deadline should have the precedence and win the arbitration. That is, if $d_i < d_j$, then the packet of f_i will be able to preempt the packet of f_j, and vice versa.

What are the requirements to implement such a policy? With respect to the router's logic, very few changes are needed; instead of packet priorities, packet deadlines are compared. However, notice that when a packet is tagged with its deadline, that value highly depends on the perception of time of the processor releasing it. Due to the finite signal propagation speed, different temperatures and different physical compositions, it is quite common that two components of the same system receive the same signal at different time instants. This infers that two processors may perceive the same time instant at two different moments, which is in the circuit design theory called the *clock skew*. Ultimately, the clock skew is inevitable and chip manufacturers employ various techniques to mitigate its effects, however, that topic is beyond the scope of this work. Here, we just assume that the parameter Δ denotes the maximum clock skew. The implications are explained with the following example. Consider that, in a hypothetical case, two processors release two packets with the identical deadline D, one at the absolute time $t + D$ and the other slightly later

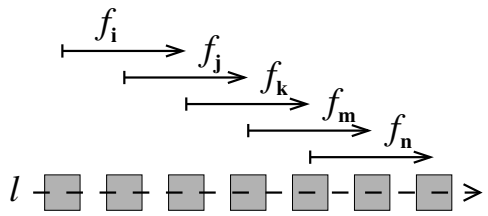

Figure 4: Example of Contending Flows

at $t + D + \epsilon$, where $\epsilon < \Delta$. Due to the clock skew, the latter processor may still tag its packet with the deadline value which is less than the one with which the former processor tagged its packet. Consequently, if these packets contend, the latter will win the arbitration, despite the fact that it was indeed released ϵ time units after the former. This effect has to be considered in the analysis.

5.2 Worst-Case Analysis

As already described (Section 4.2), the worst-case traversal time of a flow f_i consists of several components, namely the isolation delay C_i, the lower-priority blocking B_i and the higher-priority interference I_i. The first two terms were made independent of the arbitration policy. In order to be able to test the schedulability of the flow-set, we need to obtain the interference component.

Consider the example illustrated in Figure 4 with the flow characteristics given in Table 2. Let us try to straightforwardly apply the analysis for an EDF-scheduled uniprocessor system with implicit deadlines. For such a model, both the necessary and sufficient condition for schedulability is that the total system utilisation does not exceed the value of one [12], i.e. $U \leq 1$. In Section 4.2 we mentioned that the NoC contention analysis is performed in such a way that a path of a flow under analysis is treated as an indivisible resource, and any request for any of its parts by other flows is considered as the potential interference. This implies that, in our example, we have to check if the utilisation of the path of each flow, treated as a uniprocessor, fulfils the schedulability condition, i.e. $\forall f_x \in \{f_i, f_j, f_k, f_m, f_n\} : U_x \leq 1$.

Table 2: Example of flow characteristics

Flow	C	B	D = T
f_i	3	0	9.98
f_j	3	0	9.99
f_k	1	0	7
f_m	6.02	0	11.01
f_n	3	0	10

$$U_i = \frac{C_i + B_i}{T_i} + \frac{C_j + B_j}{T_j} \approx 0.6$$

$$U_j = \frac{C_i + B_i}{T_i} + \frac{C_j + B_j}{T_j} + \frac{C_k + B_k}{T_k} \approx 0.74$$

$$U_k = \frac{C_j + B_j}{T_j} + \frac{C_k + B_k}{T_k} + \frac{C_m + B_m}{T_m} \approx 0.99$$

$$U_m = \frac{C_k + B_k}{T_k} + \frac{C_m + B_m}{T_m} + \frac{C_n + B_n}{T_n} \approx 0.99$$

$$U_n = \frac{C_m + B_m}{T_m} + \frac{C_n + B_n}{T_n} \approx 0.85$$

The results suggest that the flow-set is schedulable. However, Figure 5 demonstrates that missed deadlines may occur! The explanation is identical to that of the fixed-priority example; indirect interferences cause jitters, which were not considered in the analysis and yet have an impact on the

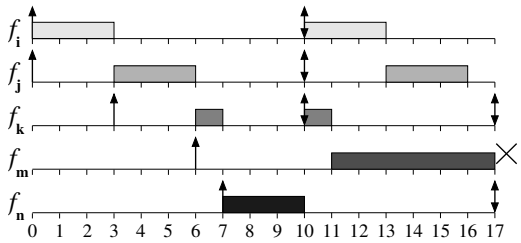

Figure 5: Deadline Miss Example

schedulability. In this particular example, f_j indirectly interferes with f_m in a sense that it causes the jitter to f_k, which eventually leads to a missed deadline of f_m. Thus, in the presence of jitters, having the utilisation of all paths less than or equal to 1 is not a sufficient condition for a flow-set to be schedulable. In order to be able to test the schedulability of a flow-set, we have to find a way to include jitters in the analysis.

In the scheduling theory, the model that bears the closest resemblance to ours is the one that involves EDF-scheduled uniprocessor systems with release jitters. Spuri [20] proposed the analysis for such a model. He defined a *busy period*, which represents the time interval of maximal length with a continuous execution demand, assuming that the first jobs of all tasks were released with the maximum jitters. Subsequently, he proved that if no missed deadlines occur within the busy period, the system is schedulable.

Recall (Section 4.2), that in the NoC contention analysis, a path of each flow is treated as a uniprocessor. Thus, unlike in the scheduling theory where only one busy period is computed, here we have to compute one for each flow (path). For now, let us assume that the jitters are known, and later we will explain how to compute them. The length of the busy period W_i (Equation 6) is obtained by summing up the maximum load that can be generated by the flow under analysis f_i, and the maximum load that can be generated by all the other flows that share a part of the path with f_i.

$$W_i = \left\lceil \frac{W_i + J_i}{T_i} \right\rceil \times (C_i + B_i) + \sum_{\forall f_j \in path(f_i)} \left\lceil \frac{W_i + J_j}{T_j} \right\rceil \times (C_j + B_j) \tag{6}$$

Once the busy period has been computed, we have to check whether the analysed flow f_i can miss a deadline during that period. A set of time instants, for which we need to check, are those where the deadlines of f_i and at least one of the potentially interfering flows coincide, i.e. $\mathcal{T} = \bigcup_{\forall f_j \in path(f_i)} \{k * T_j - T_i, k \in \mathbb{N}_0\} \cap [0, W_i]$, hereafter referred to as the *critical instants*. Assuming that a packet is released at the critical instant $t \in \mathcal{T}$, its worst-case traversal time $L_i(t)$ (expressed in absolute values) is computed by solving Equation 7.

$$L_i(t) = \left(1 + \left\lfloor \frac{t + J_i}{T_i} \right\rfloor\right) \times (C_i + B_i) + \sum_{\substack{\forall f_j \in path(f_i) \\ T_j \leq t + T_i + J_j + \Delta}} \min\left\{ \left\lceil \frac{L_i(t) + J_j}{T_j} \right\rceil, \left\lfloor \frac{t + T_i + J_j + \Delta}{T_j} \right\rfloor \right\} \times (C_j + B_j) \tag{7}$$

Equation 7 consists of the sum of the latencies of all packets of f_i, which were released in the interval $[0 - L_i(t)]$, augmented by the interference that the packets of f_i may suffer from the packets of other flows. Individual terms are obtained by finding the smaller between (i) the maximum number of releases of an interfering flow within the interval

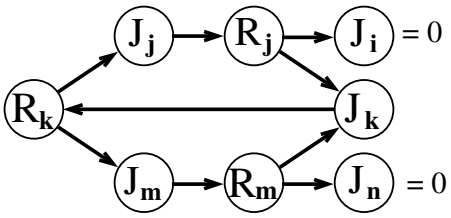

Figure 6: Chain of dependencies for Figure 4

$[0 - L_i(t)]$, and (ii) the maximum number of those releases which deadline falls before, or coincides with $L_i(t)$. Note, Equation 7 is similar to the one Spuri [20] proposed for an EDF-scheduled uniprocessor system with jitters. The differences are that our approach considers the maximum clock skew (parameter Δ) and implicit deadlines.

The worst-case traversal time of a packet released at the critical instant t, expressed in relative terms, is computed by subtracting its release t from the obtained value $L_i(t)$. The additional remark is that the result cannot be less than $C_i + B_i$, i.e. $R_i(t) = \max\{C_i + B_i, L_i(t) - t\}$. Finally, upon obtaining the traversal times for all the critical instants, we compute the worst-case traversal time of a flow f_i by finding the maximum (Equation 8). Of course, both the necessary and sufficient schedulability condition is $R_i \leq D_i = T_i$.

$$R_i = \max\{R_i(t), \forall t \in \mathcal{T}\} \tag{8}$$

5.3 Jitter Computation

The analysis proposed in the previous section is applicable under the assumption that jitters of all flows are known in advance. Recall, jitters are the way to model the effects of indirect interferences, and have non-zero values only for those directly competing flows which are involved in the indirect interference relationship (from the perspective of the analysed flow). Thus, it is trivial to see that the jitter of the analysed flow is always zero, i.e. in Equations 6-7 $J_i = 0$.

Recall, that in the analysis for the fixed-priority model (Section 4.3), it is safely assumed that each contending flow that is involved in the indirect interference relationship (from the perspective of the analysed flow) experiences the maximum possible jitter, while still being schedulable, i.e. $J_i = R_i - C_i$. That approach was straightforwardly applicable, because in the fixed-priority model the commutative property applied to preemptions does not hold. In other words, for any two contending flows f_i and f_j it holds that, if $P_i > P_j$, then f_i can preempt f_j, but f_j cannot preempt f_i. Thus, the existence of f_j has no effect on the analysis of f_i and the worst-case traversal time and the jitter can be computed first for f_i and then for f_j. In fact, by sorting the flows decreasingly by their priorities, and by performing the analysis in that order, the worst-case traversal times and jitters of all flows can be obtained in a single pass.

Conversely, in the model with the EDF arbitration policy, the commutative property applied to preemptions may hold, that is, two contending flows f_i and f_j may preempt each other. This infers that if we apply the aforementioned approach to compute jitters ($J_i = R_i - C_i$), we may encounter a circular dependency. We show this with an illustrative example. Consider the flow-set from Figure 4 and let us try to compute the worst-case traversal time of the flow f_k. It is visible that R_k depends on both J_j and J_m, which both have non-zero values due to the existence of f_i and f_n, respectively. Both jitters J_j and J_m depend on the respective worst-case traversal times R_j and R_m. Furthermore, R_j depends on J_i and J_k. J_i has a value of zero, because f_i does not have directly competing flows which may cause indirect interference to f_j. However, J_k has a non-zero value, due to the existence of f_m. J_k depends on R_k. Notice, that we have reached the computation of R_k for the second time, which infers that we encountered a circular dependency. And indeed, if we construct the chain of dependencies for the given example (Figure 6), we can see that $R_k \rightarrow J_j \rightarrow R_j \rightarrow J_k \rightarrow R_k$, and $R_k \rightarrow J_m \rightarrow R_m \rightarrow J_k \rightarrow R_k$ are circles of dependencies.

Algorithm 1: Comp_WCTT_All_Flows($\mathcal{F}$)

```
   input : F
1  foreach (f_i ∈ F) do
2      R_i = C_i + B_i;// New WC traversal time
3      R_i^* = 0;// Old WC traversal time
4  end
5  while (∃f_i ∈ F : R_i^* ≠ R_i) do
6      foreach (f_i ∈ F) do
7          R_i^* = R_i;
8          R_i = Comp_WCTT_Flow(f_i, F);
9          if (R_i > D_i) then
10             return unschedulable;
11         end
12     end
13 end
14 return schedulable;
```

As discussed above, the commutative property with respect to preemptions may hold for EDF-arbitrated NoCs, and may cause circular dependencies. Therefore, jitters and the worst-case traversal times cannot be straightforwardly computed like it the fixed-priority model. We propose to solve this problem in an iterative way. Algorithms 1-2 describe how to compute jitters and the worst-case traversal times, in an interleaved fashion. The computation process starts by invoking the function described with Algorithm 1. For each flow f_i of the flow-set we keep the previously and the newly computed value of the worst-case traversal time, R_i^* and R_i, respectively. Initially, we start by assuming that the worst-case traversal time of each flow is equal to its isolation latency, augmented by the lower-priority blocking (line 2). Then, for each flow, we invoke the function described with the Algorithm 2, which computes the worst-case traversal time of a flow (line 8). While there exists at least one flow for which the new and the old value of the worst-case traversal times are different, the process is repeated (line 5). Finally, when the worst-case traversal times from two successive iterations are equal for every flow, the computation process terminates. That is, the stopping condition is: $\forall f_i \in \mathcal{F} : R_i = R_i^*$. If, at any stage, the computed worst-case traversal time of any flow exceeds its deadline, the entire flow-set is rendered unschedulable (lines 9-11).

The computation of the worst-case traversal time of an individual flow is described with Algorithm 2. The algorithm has three stages. The first one generates a list $\mathcal{F}_i$, which contains all flows contending with the analysed flow f_i (lines 1-6). Then, for each flow $f_j \in \mathcal{F}_i$, it is tested whether it has a contending flow f_k which may indirectly interfere with f_i. If at least one such flow exists, the jitter J_j of the flow f_j has a non-zero value and is computed by subtracting its isolation latency from its worst-case traversal time (lines 8-10). Conversely, if there exists no f_k which can cause indirect interference to f_i through f_j, it follows that $J_j = 0$ (lines 11-13). Once the jitters of all contending flows are obtained,

Algorithm 2: Comp_WCTT_Flow(f_i, $\mathcal{F}$)

```
   input : f_i, F
   // 1. Find all flows contending with f_i
 1 F_i = ∅;
 2 foreach (f_j ∈ F : f_j ≠ f_i) do
 3     if (path(f_j) ∩ path(f_i) ≠ ∅) then
 4         add (F_i, f_j);
 5     end
 6 end
   // 2. Compute jitters of all contending flows
 7 foreach (f_j ∈ F_i) do
 8     if (∃f_k ∈ F : f_k ∉ F_i ∧ path(f_k) ∩ path(f_j) ≠ ∅) then
 9         J_j = R_j − C_j;
10     end
11     else
12         J_j = 0;
13     end
14 end
   // 3. Compute the WC traversal time of f_i
15 W_i = Comp_Busy_Interval(f_i ∪ F_i); // Equation 6
16 T = Find_Crit_Pts(f_i ∪ F_i, W_i); // Section 5.2
17 R_i = C_i + B_i;
18 foreach (t ∈ T) do
19     L_i(t) = Comp_Abs_Time(f_i ∪ F_i, t); // Equation 7
20     R_i = Max(R_i, L_i(t) − t);
21 end
22 return R_i;
```

R_i is computed from Equations 6-8 (lines 15-22).

5.4 Discussion

In the previous two sections we have proposed the analysis to compute the worst-case traversal time of a flow, assuming the EDF arbitration policy. Consequently, if it holds that $\forall f_i \in \mathcal{F} : R_i \leq D_i = T_i$ the entire flow-set is schedulable.

In Section 4 we have indicated that, despite the fact that the NoC contention analysis is similar to the uniprocessor scheduling theory, there are some differences as well, which have several interesting implications. For instance, Shi and Burns [16] showed that the rate-monotonic priority assignment technique is not optimal in the NoCs context where flows have fixed priorities, while the opposite is the well-known fact in the uniprocessor scheduling theory [12]. Similarly, it is well-known that EDF is the optimal scheduling policy for uniprocessors [12], so it will be interesting to investigate if there exist cases in the NoC context where the fixed-priority arbitration policy outperforms EDF. The following two case studies further explore these ideas.

5.4.1 Case-study 1 (EDF outperforms FP)

Consider the example of only two contending flows, with the characteristics given in Table 3. Since only two flows are involved, there are no indirect interferences, i.e. both jitters J_i and J_j are zero. Let us first try to test the schedulability of the flow-set, assuming that the priorities have been assigned in the rate-monotonic fashion: $T_i < T_j \Rightarrow P_i > P_j$.

Table 3: Example of flow characteristics

Flow	C	B	D = T
f_i	5	0	10
f_j	6	0	15

$$R_i = C_i + B_i = 5 < T_i$$

$$R_j = C_j + B_j + \left\lceil \frac{R_j + J_i}{T_i} \right\rceil \times (C_i + B_i) = 16 > T_j$$

The flow f_j is unschedulable. Now let us test the schedulability if the priorities are assigned differently: $P_j > P_i$.

$$R_j = C_j + B_j = 6 < T_j$$

$$R_i = C_i + B_i + \left\lceil \frac{R_i + J_j}{T_j} \right\rceil \times (C_j + B_j) = 11 > T_i$$

In this case the flow f_i is unschedulable. Let us now test the schedulability of this flow-set assuming the EDF arbitration policy. Since both jitters are zero, it is sufficient to only test if the utilisations of $path(f_i)$ and $path(f_j)$ are less than one.

$$U_i = U_j = \frac{C_i + B_i}{T_i} + \frac{C_j + B_j}{T_j} = 0.9$$

As $U_i = U_j < 1$, the flow-set is schedulable.

5.4.2 Case-study 2 (FP outperforms EDF)

Consider the example of flows given in Figure 2, with the flow characteristics given in Table 4. By applying the rate-monotonic priority assignment policy it follows that $P_i > P_j$ and $P_k > P_j$, thus, there are again no indirect interferences and jitters. Let us test the schedulability of this flow-set.

Table 4: Example of flow characteristics

Flow	C	B	D = T
f_i	2	0	6
f_j	3	0	7
f_k	2	0	6

$$R_i = C_i + B_i = 2 < T_i$$

$$R_k = C_k + B_k = 2 < T_k$$

$$R_j = C_j + B_j + \sum_{\forall f_m \in \{f_i, f_k\}} \left\lceil \frac{R_j + J_m}{T_m} \right\rceil \times (C_m + B_m) = 11 > T_i$$

The flow f_j is unschedulable. Now let us test the schedulability of the flow-set when the priorities are assigned differently: $P_j > P_i \wedge P_j > P_k$. It is easy to see that again indirect interferences do not exist, and hence jitters are equal to zero.

$$R_j = C_j + B_j = 3 < T_j$$

$$R_i = C_i + B_i + \left\lceil \frac{R_i + J_j}{T_j} \right\rceil \times (C_j + B_j) = 5 < T_i$$

$$R_k = C_k + B_k + \left\lceil \frac{R_k + J_j}{T_j} \right\rceil \times (C_j + B_j) = 5 < T_k$$

The flow-set is now schedulable. Finally, let us test if the flow-set is also schedulable with the EDF arbitration policy. Notice that in this case indirect interferences do exist, i.e. when f_i is under analysis, then f_k can indirectly interfere, and vice versa. Let us first try to compute the busy period for the flow f_j.

$$W_j = \left\lceil \frac{W_j + J_j}{T_j} \right\rceil \times (C_j + B_j) +$$

$$\sum_{\forall f_m \in \{f_i, f_k\}} \left\lceil \frac{W_j + J_m}{T_m} \right\rceil \times (C_m + B_m) \rightarrow \infty$$

As the busy period cannot be computed, the flow-set is unschedulable. The same conclusion can be reached by computing the utilisation of $path(f_j)$, because $U_j \approx 1.095$.

These two case studies have shown that there are scenarios where the EDF arbitration policy can schedule a flow-set, while no priority assignment exists which can cause the fixed-priority arbitration policy do the same (the first row in Table 5). Similarly, we have shown that the opposite is

true as well, and also have confirmed the findings of Shi and Burns [16] that there exists a priority-assignment which can schedule the flow-set which is unschedulable with priorities assigned in the rate-monotonic fashion (the second row in the Table 5). The case-studies, for which the third and the fourth row of Table 5 are true, are omitted due to space constraints. Finally, it is trivial to see that there exist flow-sets for which the fifth and the sixth row are true. This brings us to the second major difference between the uniprocessor scheduling theory and the NoC contention analysis. Although in the former EDF is proven to be optimal [12], in the latter a flow-set can be schedulable with the fixed-priority arbitration policy, but not with EDF (see Case-study 2).

Table 5: Comparison of approaches (schedulability)

EDF	Rate-monotonic	Exists priority assignment
✓	X	X
X	X	✓
✓	X	✓
X	✓	✓
✓	✓	✓
X	X	X

6. EXPERIMENTS

In this section we evaluate the proposed analysis for EDF-arbitrated NoCs, in the further text referred to as the *EDF method*, against the two existing state-of-the-art approaches for FP-arbitrated NoCs [16, 17]. In the first, the priorities are assigned in the rate-monotonic fashion. In the second, the heuristics-based search algorithm (HSA) is used to find a priority ordering, if one exists, such that the flow-set is schedulable. We refer to these methods as to *RM* and *HSA*, respectively. Although HSA is based on the heuristics, it has one limitation: if it is unable to find a priority ordering with which the flow-set is schedulable, it will exhaustively enumerate all possible priority orderings. This infers that HSA has a factorial computational complexity (i.e. for flow-sets with $|\mathcal{F}|$ flows there exist $|\mathcal{F}|!$ different priority orderings), which further implies that HSA can be inapplicable in scenarios where flow-sets consist of 50 or more flows. Thus, for HSA we impose the limit on the maximum number of orderings that can be evaluated. Specifically, for the flow-set of $|\mathcal{F}|$ flows, we allow HSA to attempt at most $5 \times |\mathcal{F}|$ different priority orderings. If HSA fails to find an ordering with which the flow-set is schedulable, the process terminates.

6.1 Evaluation Metrics and Parameters

The comparison of the approaches is performed through the *sensitivity analysis* with respect to flow sizes. Specifically, if a flow-set is unschedulable with initial flow sizes, then the sizes of all flows are uniformly decreased until the flow-set becomes schedulable. Similarly, if a flow-set is schedulable with initial sizes, then the sizes of all flows are uniformly increased until the flow-set becomes unschedulable. The maximum flow sizes for which one method can guarantee the schedulability of a flow-set is called the *schedulability threshold* (ST). Of course, a higher ST infers that the method is more efficient. Upon obtaining the STs for the same flow-set with all the approaches, we compare them. Let ST_{EDF}, ST_{RM} and ST_{HSA} be the STs obtained for the EDF method and with the two state-of-the-art methods. We measure the improvements of the proposed approach over the existing ones in the following way: $Imp_{EDF/RM} = \frac{ST_{EDF} - ST_{RM}}{ST_{RM}}$, and $Imp_{EDF/HSA} = \frac{ST_{EDF} - ST_{HSA}}{ST_{HSA}}$.

The flow-set and analysis parameters are summarised in Table 6, where an asterisk sign denotes a randomly generated value assuming a uniform distribution.

Table 6: Analysis and flow-set parameters

Platform size	**8×8**
NoC frequency	**2GHz**
Router latency + link latency	**3 + 1 cycles**
Link width = flit size	**16B**
Flow size	**[1 - 128]* KB**
Flow periods	**[20 - 100]* μs**

6.2 Experiment 1: Overall improvements

In this experiment we observe the improvement trends with respect to flow path lengths. We do that by imposing a constraint that the maximum path length (expressed in hops), of any flow of the flow-set, cannot exceed the value of the newly introduced parameter LIM, i.e. $\forall f_i \in \mathcal{F} : nhops(f_i) \leq LIM$. We vary the parameter LIM in the range $[1 - 14]$. $LIM = 1$ means that only single-hop paths are allowed. Conversely, $LIM = 14$ allows all possible paths, because, assuming the XY routing, the maximum path length on a 8×8 platform is 14 hops long. For each value of the parameter LIM we randomly generate 1000 flow-sets, each consisting of 200 flows. For each flow, the initial size and the period are randomly generated, while the source and the destination router were generated in a way that the path length does not exceed the imposed limit LIM. Subsequently, we compute ST_{EDF}, ST_{RM} and ST_{HSA} of each flow-set, and compare the obtained values.

Figure 7 shows the improvements of EDF over RM. It is visible that, for cases where $LIM = 1$, EDF dominates RM. The explanation is as follows. When $LIM = 1$ the transitivity property always holds, and consequently indirect interferences do not exist. This infers that the rules from the uniprocessor scheduling theory also hold in this context: (i) EDF is the optimal policy and hence always outperforms RM, and (ii) RM is optimal among fixed-priority policies. On the rest of the domain ($LIM > 1$), the transitivity property may not hold, which implies that indirect interferences are possible. Thus, as exhibited in Case Studies 1-2, there exist cases where EDF outperforms RM, but the opposite is also true. Yet, the cases in which EDF performs better are more frequent, and on average EDF outperforms RM by 7%. As LIM grows, the average improvements also grow, however, the increase is barely noticeable.

Figure 8 shows the improvements of EDF over HSA. We can see that for $LIM = 1$ EDF also dominates HSA, in fact, the improvements are the same as in the example with RM. This is expected, because, as stated in the previous paragraph, RM is optimal among fixed-priority policies, and hence it should be HSA = RM. Thus, we can conclude that EDF presents a very promising approach for systems where flows mostly traverse single-hop distances. On the rest of the domain ($LIM > 1$) HSA significantly outperforms EDF, and as the lengths of flow paths increase, the dominance of HSA becomes more apparent. This is a very counter-intuitive finding and the explanation is as follows. In EDF the commutative property holds, and in order a flow-set to be schedulable, the necessary condition is that the utilisation of the path of each flow is less than or equal to 1. Conversely, in the fixed-priority scheme, the commutative property does not hold, and the flow can be schedulable even though the utilisation of its path is greater than 1 (see Case Study 2). In fact, we can conclude that, when $LIM > 1$, almost al-

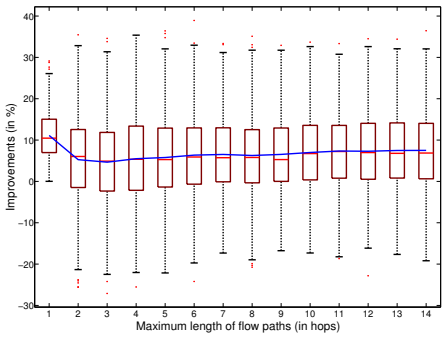

Figure 7: EDF vs RM

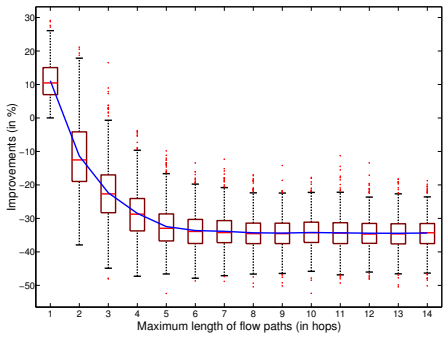

Figure 8: EDF vs HSA

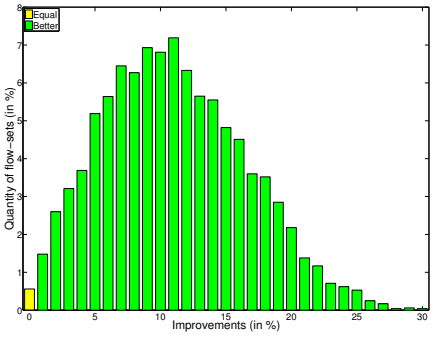

Figure 9: EDF vs RM,HSA (1-hop)

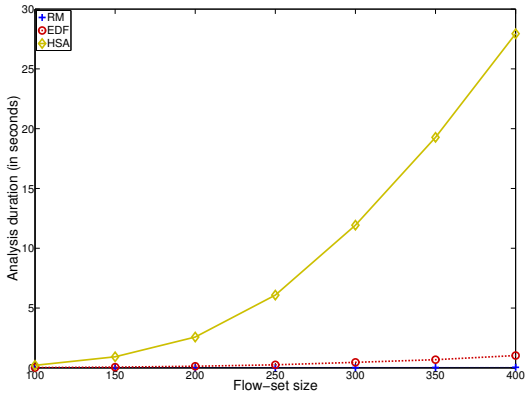

Figure 10: Influence of flow-set size on analysis time

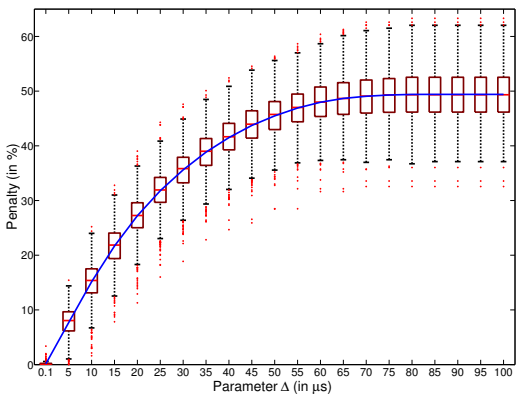

Figure 11: Influence of Δ on EDF

ways there exists a priority ordering that can schedule a flow-set which is unschedulable with EDF. This is a crucial finding which suggests that EDF (or any other arbitration policy with dynamically changing flow priorities) may not be the most efficient arbitration technique for flow-sets with arbitrary path lengths. Yet, before this can be discussed, another important question has to be answered: how hard it is to find a priority ordering which will succeed in cases where EDF fails? This is covered in Experiment 2.

Assuming that all flows traverse single-hop distances, Figure 9 illustrates the improvements of EDF over the optimal fixed-priority scheme (i.e. RM). We used the same data as in Figures 7-8, but we focused only on $LIM = 1$. It is visible that the average improvements are around 10%, while in some cases can reach up to 30%.

6.3 Experiment 2: Scalability

In this experiment we observe how the analysis duration time changes with the increase in the flow-set size. In other words, we want to observe how scalable are EDF, RM and HSA. We vary the number of flows constituting the flow-set in the range $[100 - 400]$, with an incremental step of 50 flows. For each flow-set size we randomly generate 1000 flow-sets. For each flow the initial size, the period, the source and the destination routers are randomly generated, without the maximum path length constraint, i.e. $LIM = 14$. For each flow-set we measure the time it takes to compute ST_{EDF}, ST_{RM} and ST_{HSA} on an Intel Pentium dual-core platform. Subsequently, we compare the obtained values.

Figure 10 demonstrates that the computational complexity of EDF is higher than that of RM. This is expected, because in RM the commutative property does not hold, and hence the worst-case traversal times and the jitters can be obtained in a single pass, if flows are ordered by their priorities, decreasingly. Conversely, in EDF the commutative property holds, and hence several passes are needed until the worst-case traversal times and jitters stabilise for two successive passes (see Section 5.3 and Algorithms 1-2).

In spite of imposing the limit on the maximum number of orderings in HSA to only $5 \times |\mathcal{F}|$, its computational complexity significantly surpasses that of EDF and RM, and the duration time of the analysis grows exponentially. This happens because HSA exhaustively enumerates the maximum allowed number of orderings before eventually rendering the flow-set unschedulable, while RM and EDF reach that conclusion much faster. These findings imply that HSA is the least scalable of the compared approaches, and that searching for ST_{HSA} may be prohibitively expensive for flow-sets with more than 500 flows. Therefore, we can conclude that, for massive flow-sets, RM and EDF are preferable options.

6.4 Experiment 3: Applicability

The common underlying assumption of the previous experiments is that the clock skew does not exist, i.e. $\Delta = 0$ in Equation 8. In this experiment we want to quantify the sensitivity of the EDF method with respect to the clock skew. Based on the findings, we will be able to derive some conclusions regarding the practical limitations of the EDF arbitration policy. The experiment is conducted in the following way. First, through the sensitivity analysis, we obtain ST_{EDF} of one flow-set, when the clock skew is equal to zero. That value serves as a baseline for comparisons. Then, for a non-zero value of the clock skew we again perform the sensitivity analysis and obtain $ST_{EDF'}$ of the same flow-set. We compare these values and express the penalty suffered due to the clock skew with the following

metric: $penalty = \frac{ST_{EDF} - ST_{EDF'}}{ST_{EDF}}$. We repeat that process for 1000 flow-sets with the fixed flow-set size $|\mathcal{F}| = 200$ and $LIM = 14$. Subsequently, we change the clock skew value ($\Delta \in [0.1 - 100\mu s]$) and compute the penalty again.

The results are depicted in Figure 11. It is noticeable that for $\Delta \leq 0.1\mu s$ the effects are negligible. In the range $0.1\mu s < \Delta \leq 100\mu s$ the penalty grows logarithmically until reaching a saturation point at $\Delta = 80\mu s$. The conclusion is that the schedulability is very sensitive to the values of the clock skew that are within the range of flow periods, which is intuitive. It is also interesting that the saturation point exists. The explanation is that, even in the hypothetical case, with the infinite clock skew, the interference that one flow can cause to another ultimately has an upper-bound which is equal to the maximum number of releases of the interfering flow within the interval of interest (the first term in the min function of Equation 8). Regarding the most important question about the applicability of EDF, we can conclude that the clock skew values that have an impact on the schedulability are several orders of magnitude greater than the ones in the real systems, inferring that EDF, counter-intuitively, does not face practical limitations.

7. CONCLUSIONS

The contention analysis of the interconnect mediums of multiprocessors is still in its infancy. Motivated with the facts from the scheduling theory, this work is an initial attempt to consider dynamically changing traffic flow priorities for wormhole-switched, priority-preemptive NoCs. Our primary objective is to investigate whether such approaches can utilise the interconnect resources more efficiently than the existing methods which rely on fixed flow priorities. In particular, we explored the possibility to use the EDF paradigm as an arbitration policy. We elaborated on the prerequisites for enforcing such an arbitration policy. Then, we proposed the analysis to compute the worst-case traversal times of individual traffic flows. Finally, we investigated the practical limitations of the approach and also compared it against the state-of-the-art approaches (RM and HSA), which assume NoC routers with the fixed-priority arbitration policy [17].

The experiments demonstrated that EDF dominates all fixed-priority schemes in cases where traffic flows traverse single-hop distances. In such cases the system inherits the properties of the uniprocessor scheduling theory, where EDF is dominant, and where the system resources can be utilised in the most efficient way (even up to 100%). Can this finding be exploited on our quest towards efficient and real-time oriented multiprocessors? We argue that, in order to provide an answer, further research in the area of application mapping is needed, because flow paths are inevitably dependant on the position of the tasks within the platform.

For longer flow paths, there are cases where EDF performs better than RM, but the opposite is also true. However, the number of cases where EDF performs better than RM are more frequent, and, on average, EDF outperforms RM by 7%. Moreover, HSA systematically outperforms EDF for all cases where path lengths exceed 3 hops. This finding suggests that, for flow-sets with arbitrary path lengths, EDF (or any other arbitration scheme with dynamically changing flow priorities) may not be the most efficient arbitration policy, which is an important but negative finding. However, the experiments also suggest that searching for the priority ordering that outperforms EDF indeed can be prohibitively expensive. Therefore, the applicability of EDF to a specific flow-set highly depends on the parameters of the flow-set.

Finally, the experiments demonstrated that EDF does not suffer practical limitations with respect to the clock skew. Specifically, the values of the clock skew, for which the analysis becomes sensitive, exceed the clock skews in the real systems by several orders of magnitude.

8. REFERENCES

[1] S. Baruah and T. Baker. Schedulability analysis of global edf. *Real-Time Syst. J.*, 2008.

[2] L. Benini and G. De Micheli. Networks on chips: a new soc paradigm. *The Comp. J.*, 35(1):70 –78, jan 2002.

[3] W. Dally. Virtual-channel flow control. *Trans. Parall. & Distr. Syst.*, 3(2):194 –205, Mar 1992.

[4] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *Trans. Computers*, 1987.

[5] D. Dasari, B. Nikolić, V. Nelis, and S. M. Petters. Noc contention analysis using a branch and prune algorithm. *Trans. Emb. Comput. Syst.*, 2013.

[6] T. Ferrandiz, F. Frances, and C. Fraboul. A method of computation for worst-case delay analysis on spacewire networks. In *IEEE Int. Symp. Industrial Emb. Syst.*, 2009.

[7] T. Ferrandiz, F. Frances, and C. Fraboul. Using network calculus to compute end-to-end delays in spacewire networks. *SIGBED Rev.*, 2011.

[8] J. Hu and R. Marculescu. Energy-aware mapping for tile-based noc architectures under performance constraints. In *8th ASPDAC*, 2003.

[9] Intel. *Single-Chip-Cloud Computer*. `www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/intel-labs-single-chip-cloud-article.pdf`.

[10] N. K. Kavaldjiev and G. J. M. Smit. A survey of efficient on-chip communications for soc. In *4th Symp. Emb. Syst.*, 2003.

[11] B. Kim, J. Kim, S. Hong, and S. Lee. A real-time communication method for wormhole switching networks. In *1998 Int. Conf. Parall. Processing*, Aug 1998.

[12] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *J. ACM*, 20:46–61, 1973.

[13] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *The Comp. J.*, 26, 1993.

[14] B. Nikolić, H. I. Ali, S. M. Petters, and L. M. Pinho. Are virtual channels the bottleneck of priority-aware wormhole-switched noc-based many-cores? In *21th RTNS*, 2013.

[15] Y. Qian, Z. Lu, and W. Dou. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In *Int. Symp. Netw.-on-Chip*, 2009.

[16] Z. Shi and A. Burns. Priority assignment for real-time wormhole communication in on-chip networks. In *29th RTSS*, Dec 2008.

[17] Z. Shi and A. Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Int. Symp. Netw.-on-Chip*, 2008.

[18] Z. Shi and A. Burns. Schedulability analysis and task mapping for real-time on-chip communication. *Real-Time Syst. J.*, 2010.

[19] H. Song, B. Kwon, and H. Yoon. Throttle and preempt: a new flow control for real-time communications in wormhole networks. In *1997 Int. Conf. Parall. Processing*, Aug 1997.

[20] M. Spuri. Analysis of deadline scheduled real-time systems. Technical report inria-00073920, INRIA, France, 1996.

[21] Tilera. *TILE64™ Processor*. `www.tilera.com/products/processors/TILEPro_Family`.