Time-randomized Wormhole NoCs for Critical Applications

Mladen Slijepcevic, Carles Hernandez, Jaume Abella, Francisco J. Cazorla
Barcelona Supercomputing Center (BSC)

February 19, 2019

Abstract

Wormhole-based NoCs (wNoCs) are widely accepted in high-performance domains as the most appropriate solution to interconnect an increasing number of cores in the chip. However, wNoCs suitability in the context of critical real-time applications has not been demonstrated yet.

In this paper, in the context of probabilistic timing analysis (PTA), we propose a PTA-compatible wNoC design that provides tight time-composable contention bounds. The proposed wNoC design builds on PTA ability to reason in probabilistic terms about hardware events impacting execution time (e.g. wNoC contention), discarding those sequences of events occurring with a negligible low probability. This allows our wNoC design to deliver improved guaranteed performance w.r.t. conventional time-deterministic setups. Our results show that performance guarantees of applications running on top of probabilistic wNoC designs improve by 40% and 93% on average for 4x4 and 6x6 wNoC setups, respectively.

1 Introduction

Wormhole-based NoCs (wNoCs) have been widely adopted in the high-performance computing domain as the most efficient way to connect a high number of cores within the chip. While wNoCs were adopted as a highly scalable solution to perform the interconnection of on-chip modules, the current manufacturing limitations that emanate from the utilization of smaller transistors and wires has made academia and industry start exploring emerging NoC technologies. In this regard, new wNoC approaches propose replacing metal-based links by either optical [35] or carbon-based [10] interconnections to avoid the use of power hungry metal wires. wNoCs do not only suffer from physical limitations: challenges related with the performance scalability, reliability, and security are also of prominent importance. Furthermore, with the advent of new applications requiring timing guarantees, like autonomous driving systems, wNoCs are also forced to provide good-quality performance guarantees.

In this paper we aim at allowing an efficient use of wNoCs in the context of critical-related real-time applications – such as those that can be found in aircraft, cars or trains. Critical applications require guaranteeing the functional and timing correctness of the system. To achieve these guarantees, a very thorough verification and certification process is required.

In the context ot timing correctness, the main focus of this paper, wNoCs complicate the derivation of good-quality performance guarantees since, unlike buses or other existing centralized network architectures, wNoCs perform the arbitration of communication flows in a distributed manner. In this line, some works show that, while reliable contention upper bounds can be provided for Commercial off-the-shelf (COTS) wNoCs [33] [30] [29], those bounds are pessimistic, preventing an efficient use of high-performance wNoCs for mixed-criticality real-time embedded systems (RTES).

wNoC bounds are pessimistic because, whenever timing events can lead to the stall of a request, they are assumed to occur systematically, and hence factored in the derived contention bounds. At the NoC level, since many different flows with different criticality levels might potentially be contending for different resources, e.g. router ports, timing analysis techniques are forced to make the pessimistic assumption that all contenders will simultaneously request the same resources. A simple and intuitive way to reduce contention bound pessimism consists in getting information about when and where communication flows in the wNoC will occur such that the exact interference that requests experience can be reliably and tightly factored in [37]. Unfortunately, obtaining this low-level information is not only out of the ability (and will) of end users, but it also breaks time composability. The lack of time composability occurs because one task's load on the wNoC affects the worst-case execution time (WCET) estimates of its corunners, with devastating consequences in (incremental) system integration: any change in a task requires reanalyzing all other tasks (i.e. performing regression tests), which ultimately results in prohibitively high integration costs. Even worse, the WCET of a critical task could depend on the accuracy of the information obtained for a lower criticality task, exposing critical tasks to safety and security issues.

Measurement-based probabilistic timing analysis (MBPTA) has been proposed recently as an industrially-friendly timing analysis method to derive WCET estimates and proven in bus-based multicore industrial case studies [43]. MBPTA relies on hardware designs that break systematic pathological behavior, so that increasingly high contention scenarios occur with decreasing probabilities, thus leading to low probabilistic WCET (pWCET) estimates by discarding execution times whose upper-bounded accumulated probability can be proven negligible. pWCET estimates in the context of MBPTA allow upper-bounding the residual risk of a timing fault. This does not imply that software timing faults can occur, which is not acceptable for functional safety standards, such as ISO26262 in the automotive domain [13]. Instead, MBPTA allows reducing the residual risk (quantitatively) to arbitrarily low levels (e.g. 10^{-9} per run or per hour of operation), so that above that probability no exceedance can occur. Below such probability, although the process indicates that the pWCET bound should not be exceeded, information is not practically had and must be deemed as residual risk. This information has been shown valid for certification practice of safety-related software [40].

In this paper we propose several time-randomized wNoC designs that make the contention in the network have a probabilistic behavior compatible with MBPTA requirements, thus leading to reduced contention bounds. The contributions of this paper can be summarized as follows:

1. We integrate efficiently random permutation arbitration [14] in wNoCs routers to avoid systematic bad behavior and make them amenable for probabilistic timing analysis.

2. We show that, while limiting the number of in flight requests in deterministic wNoCs[1] does not help reducing contention bounds, it helps reducing significantly those bounds in a probabilistic wNoC and thus, improves WCET estimates.

3. We propose an alternative mechanism to control the injection of packets in the wNoC based on controlling the time elapsed (frequency) between request injections rather than the number of inflight requests. The former mechanism outperforms the latter in high contention scenarios while requiring similar hardware complexity for the network interfaces.

Results obtained with a cycle-accurate simulator confirm that the proposed wNoCs achieve tighter bounds than existing wNoCs and thus, enable the derivation of much tighter WCET estimates. Contention in the NoC can be reduced significantly and that reduction becomes more significant as the size of the network increases. In particular, these improvements translate into an average reduction of the WCET estimates for EEMBC [31] workloads of 22% and 40% for networks of 9 and 16 cores, respectively. Since deterministic WCETs for larger networks are very pessimistic, we obtain up to 93.3% improvement for 36-core NoCs. Thus, our work enables efficient decentralized arbitration delivering high average and guaranteed performance, as needed in the context of mixed-criticality RTES.

2 MBPTA in the Context of wNoCs

Probabilistic timing analysis (PTA) resorts on having platforms on which the end-to-end execution time of applications can be modelled with true probabilities. Note that probabilities differ from frequencies. While frequencies provide information about past events, probabilities allow reasoning about the future and thus, make predictions. In particular, we focus on the measurement-based variant of PTA (MBPTA), since measurement-based timing analysis has been shown to be closer to industrial practice in many systems [24, 20, 25]. In this

[1] In this paper we refer to a deterministic NoC as the one with a round-robin arbitration, although similar conclusions are obtained with other time-deterministic policies.

section we review some of the key elements of MBPTA for its reliable application for WCET estimation.

2.1 MBPTA Application Process

MBPTA relies on collecting a number of execution time measurements – typically in the order of few hundreds – of the program under analysis on top of a MBPTA-compliant hardware/software platform [17]. The MBPTA method defines a convergence criteria to determine the actual number of measurements needed [9, 4]. For a sound application of MBPTA, execution time variability must be caused by random events only. Hence, sources of execution time variation need to be either removed (i.e. upper-bounded) or randomized. Non-random time variability, if it does not occur with *exactly* the same frequencies as during operation, cannot be properly modelled by MBPTA. Such degree of control has been regarded as unaffordable in practice [8].

MBPTA-compliant platforms allow collecting measurements on an *analysis mode*, where the different sources of execution time variation match or upper-bound the behavior during operation, which is achieved by applying techniques like time upper-bounding and time randomization to each individual source of execution time variation. For instance, in the case of a shared bus to access a second level (L2) cache in a multicore, it has been proven that its impact on execution time can be probabilistically upper-bounded by: (1) enforcing maximum contention and (2) granting access to one of the contenders randomly [14]. In particular, maximum contention is enforced by always arbitrating across all contender cores, regardless of whether they have pending requests and, whenever a core different to the one where the task under analysis runs is granted access, it keeps the bus busy for the maximum duration of a request. Then, such constraint can be released during operation, thus allowing the task analyzed to contend only with the actual subset of contenders with ready requests, and experience the contention caused by the duration of the contenders' requests, which may not be the maximum duration. Arbitration needs to be done randomly at analysis and during operation using the same (random) policy so that the distribution experienced at analysis matches the worst distribution that can occur during operation. Note that by operating this way, any scenario that can occur during operation – including that where all contenders request access to the bus sustainedly – is upper-bounded by the contention enforced at analysis. This allows collecting measurements in isolation representative of the worst-case multicore behavior.

Since execution times collected at analysis correspond to a random variable (variability is random and the random variable matches the worst probabilistic behavior possible during operation) and real-time programs have a finite execution time, execution time measurements must necessarily fulfill the following requirements: (1) the upper tail of execution time distributions can be modelled with a Gumbel distribution (exponential tail), which is the type of distribution upper-bounding any distribution with a maximum value (even if such value is unknown). (2) The collected execution time sample needs to attain statisti-

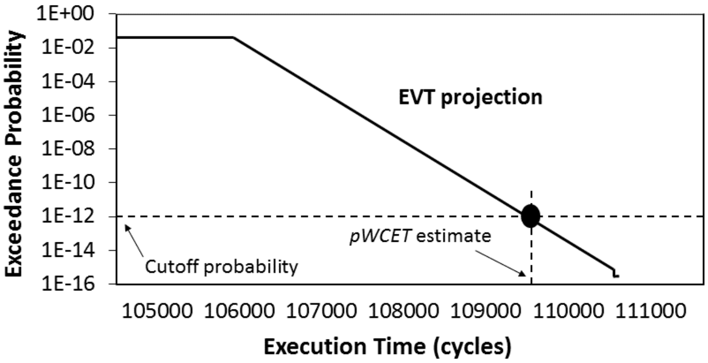


Figure 1: EVT projection (i.e. probabilistic WCET)

cal independence and identical distribution (i.i.d.). Execution times are i.i.d. probabilistically, but the sample might fail i.i.d. statistical tests. However, by increasing the sample size, it will eventually converge to the real distribution, so the sample will pass the tests. Both requirements, namely exponential tail and i.i.d., are assessed empirically for the sample of execution times used for prediction with appropriate statistical tests [9, 4].

Once those tests are passed, execution time measurements are used as input for Extreme Value Theory (EVT) [18], which is a powerful statistical method to approximate the tail of a distribution. In the case of MBPTA, the tail of the distribution corresponds to high execution times. This results in a probabilistic WCET (pWCET), which is a distribution where each execution time value has an associated probability that upper bounds the real probability of one run of the program exceeding such execution time (see the example in Figure 1). The particular cutoff probability is chosen to be low enough so that it can be regarded as residual risk, in line with safety standards requirements [13].

2.2 Requirements on the wNoC

MBPTA application requires the sources of jitter (execution time variation) to be properly controlled so that they match or upper-bound operation time conditions in either a deterministic or a probabilistic way. In the context of multicore processors, this includes contention caused by other cores during operation. As discussed before, probabilistic modelling allows discarding the execution times whose accumulated probability is low enough to be regarded as residual risk. Note that, as explained before, whether any scenario can lead to those execution times during operation can neither be proven or rejected since worst-case conditions imposed during analysis may be unfeasible during operation. This is, for instance, the case of contention in the wNoC. If arbitration decisions are deterministic, the worst-case hypothetical contention scenario could occur systematically. Instead, if those decisions are randomized, worst hypothetical contention scenarios occur with (provable) low probability even if time composability is enforced by assuming that all contenders send requests at the maximum possible rate to the worst possible target node. Next, we describe the conditions under which timing measurements have to be collected so that pWCET estimates hold valid regardless of what other tasks are consolidated on the other cores. In Section 4 we describe how to randomize wNoC timing behavior.

2.3 Upper-bounding Contention in Probabilistic wNoCs

To be able to reliably apply MBPTA, we have to ensure that measurements for the flow under *analysis* are collected under contention conditions that upper-bound those that can occur during *operation* [17]. Failing to do so prevents EVT from actually capturing unobserved contention effects into the pWCET, which could therefore be optimistic. For instance, measurements collected under contention-free conditions lead to unreliable pWCET estimates since EVT cannot reason about the events (contention in the wNoC) not captured in those contention-free measurements.

Such upper-bounding can be performed deterministically or probabilistically. Upper-bounding latency deterministically only requires forcing all wNoC requests to experience the worst-possible delay [30]. To illustrate probabilistic upper-bounding, let us assume a hardware resource whose analysis-time latency can be 1 or 2 cycles with the same probability: $etd_a =< (1,2), (0.5, 0.5) >$ where the first vector corresponds to the different latencies and the second to their associated probabilities. If during operation its execution time distribution is $etd_o =< (1,2), (0.6, 0.4) >$, then etd_a probabilistically upper-bounds etd_o since the exceedance probability for any value is higher at analysis than during operation (e.g., latency of 2 cycles is exceeded with probability 0.4 during operation and 0.5 at analysis).

In case of probabilistic designs, whether one request or another is granted access upon arbitration, depends on random choices. Hence, given a particular request of the task under analysis, the particular requests that it will find in front of it causing backpressure must have been selected randomly, those requests arbitrated simultaneously have reached the particular router also based on random choices, and arbitration will be random, thus granting forward progress to a random request among contending ones. Therefore, while the worst contention scenario is practically possible, it can only occur with a given (typically low) probability, as opposed to the deterministic case, where it could occur systematically if events align temporally in specific manners. In the case of a probabilistic design, even with the same initial time alignment of events, random arbitration will lead to contention scenarios better than the worst one with relevant probability. If we multiply this effect for all requests, naturally the probability of the hypothetical worst contention occurring for all (or many) requests drops rapidly.

Therefore, probabilistic upper-bounding requires collecting measurements during the analysis phase with a wNoC whose contention is equal or higher than that occurring during operation disregarding the time alignment of re-

Table 1: Summary of the main symbols used.

Symbol	**Description**
WCD	Time-composable upper-bound to contention delay
F_i	Packet stream traversing the same source-destination route and requiring the same grade of service along the path.
H_i	Number of hops in a flow F_i
R_i^j	Router (hop) j in a flow F_i (see Figure 2(a))
r_i^k	Packet (request) k in a flow F_i
NR_i^j	Number of queues that can potentially contend for an output port that F_i is targeting at R_i^j
$\omega(i,j)$	Function that returns the index x of the worst possible destination flow F_x that starts at the hop R_i^{j+1} and reaches the worst possible destination in terms of indirect blocking of packets of flow F_i

quests, which is naturally randomized by random arbitration policies. This can be practically achieved by making all contender cores inject packets in the wNoC at the highest frequency allowed, and targeting the node that creates highest contention for the task under analysis, which is assumed to be in a particular core. If the core where the task will be run is unknown, then the core experiencing highest contention needs to be chosen to guarantee that analysis conditions reliably upper-bound operation conditions. These conditions produce the highest (probabilistic) latency for each arbitration choice and the highest (probabilistic) backpressure on the requests of the task under analysis.

In summary, probabilistic upper-bounding allows WCET to be time-composable, aka independent of the actual traffic generated by contending applications during operation.

3 Problem Formulation

Deterministic time-composable bounds in COTS wNoCs are pessimistic since we cannot make any assumption on the contending flows and, therefore, we need to assume that all flows will produce the highest interference. In this section we describe the target setup and we show why contention bounds are pessimistic on time-deterministic wNoCs.

3.1 Network Baseline

We consider a mesh network topology as it is the most common topology used in wNoCs, though the analysis presented in this section and the wNoC designs proposed in this paper are also suitable for other network topologies (e.g. torus). The symbols used in this paper are summarized in Table 1.

In our reference mesh wNoC configuration (see Figure 2), each node comprises a *PME* (Processor/Memory element) and a router that communicates with the other nodes. The PME can be either a processor core, main memory,

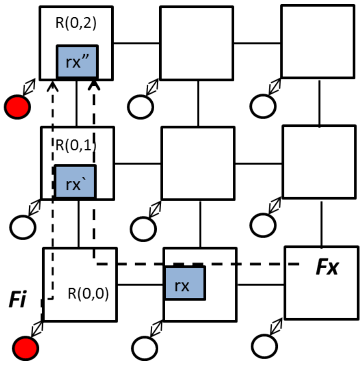


Figure 2: 3x3 Mesh.

I/O, etc. In the network, several traffic flows (F_i) may be active at the same time. Each node can be identified using (x,y) coordinates and the router located at coordinates (x,y) is referred to as $R(x, y)$.

In a wNoC, the routing algorithm determines the path that a packet follows within the network, and consequently, the number of routers or *hops* (H), a given flow requires to move from a source to a destination node. Hence, a router can also be identified as R_i^j, in which j represents the hop j of flow F_i, when moving towards its destination. Therefore, routing determines the flows that potentially contend with F_i at every router in its path. Deterministic routing has been shown to provide time analyzability [33]. We use XY routing, as it is the preferred deterministic routing algorithm for regular NoCs due to its low implementation cost, although a similar analysis is possible for any other deterministic routing policy. With XY routing, packets are forced to use the X dimension first: In the X dimension the position of the target node w.r.t. the source node determines whether to go right (X+) or left (X-) direction. The same approach is used for the Y dimension. Once a packet is routed using the Y dimension, it cannot be forwarded back to the X dimension. These routing restrictions determine the maximum number of flows contending with F_i at a given router for an output port (NR_i^j).

Communication flows comprise multiple NoC packets. A packet is the minimum arbitration unit in the network and it can be split into one or several *flits* (short of control flow units). The first flit of a packet is called header flit and contains the information required to forward the packet to the destination. We refer to the k-th packet generated by flow F_i as r_i^k. In general, one-flit packets are usually preferred for real-time workloads in order to improve wNoCs performance guarantees [29].

A typical wormhole router comprises several modules including input buffers, routing, virtual-channel allocation, switch allocation, and crossbar traversal modules. Routing modules determine the router output port based on the destination bits included in the control information of the header flit. Once the destination is known, the target port is requested in the arbitration module included in the allocation stage. Then, based on a given arbitration policy, the router arbiter decides which packet is granted access to the output port. The majority of COTS wNoCs use round-robin to arbitrate amongst packets requesting access to a given output port. Arbitration only occurs at the packet level and for the header flit. Once a connection is established between an input and output port, it remains until the tail of the packet leaves the router. At this moment a new arbitration can be performed in case other requests are also requesting this output port.

3.2 Contention in the wNoC

The latency experienced by a packet to traverse the network in the absence of contention is referred to as *zero-load latency* (*zll*). However, contention may cause the header flit to get stalled. When this happens, the remaining flits of the packet get also stalled and latency experienced by a given packet is higher than *zll*.

The first element to consider when computing the contention in the network is the number of flows that will be actually contending for the different shared resources. In our case, as we are after time-composable contention delay bounds, no assumptions can be made on the particular active flows in the wNoC. That is, it is assumed that any node in the network is entitled to send and receive packets from any other node. Similarly, when computing the contention delay for a packet, we assume that, by the time it is injected in the network, any other potential contending flow is also active at that moment, transmitting its packets in a way that it produces the worst possible contention scenario. In order to reproduce the worst possible contention scenario we need to consider the *worst direct contention* and the *worst indirect contention* [15].

Let us illustrate the process of measuring contention in the wNoC with an example. Let us consider a 3x3 wNoC setup like the one shown in Figure 2. We want to measure the worst contention experienced by a packet P_i of flow F_i. F_i is the flow originated at the node attached to $R(0,0)$ with destination the node attached to $R(0,2)$. At R_i^1, first router, a request r_x coming from port $X-$ might be potentially contending with P_i for the same output port and in the worst-case r_x will be granted access first. However, due to the backpressure flow-control it is not guaranteed that, at the time r_x is granted access, it will leave the router as the input buffer of the next router ($R(0,1)$) might be occupied. In the plot $R(0,1)$ input buffer is occupied by r'_x. In general, to measure contention it is required to iterate from the destination node backwards to analyze the time that is required by a packet to leave a given router. That is, to have a slot available in an $R(0,1)$ input buffer, we need to consider the time that r'_x requires to leave $R(0,1)$ that in turns depends on the time r''_x needs to leave the input buffer at $R(0,2)$. Equation 1 was proposed in [30] and provides a general formulation for the worst contention experienced in wNoCs under the scope of time-composability.

$$\overline{WCD_i} = \sum_{j=1}^{H_i} \left((NR_i^j - 1) \times \prod_{m=1}^{H_{\omega(i,j)}} NR_{\omega(i,j)}^m \right) \tag{1}$$

In the equation above, the first multiplicand $(NR_i^j - 1)$ corresponds to the *contention introduced by the round robin arbitration* in each of the routers that the flow F_i traverses. It represents the contention caused by the flows contending with F_i in the current router. However, in wNoCs the effect of backpressure may also prevent F_i to progress when the arbiter grants it access to the output link. This contention corresponds to the *indirect contention delay* and it is modeled by the second multiplicand $\prod_{m=1}^{H_{\omega(i,j)}} NR_{\omega(i,j)}^m$. The worst *indirect contention delay* in each hop is the one caused by the worst possible destination flow $F_{\omega(i,j)}$. The number of hops in the worst possible destination flow is $H_{\omega(i,j)}$. The worst possible destination depends on the routing algorithm as well as on the actual number of ports that routers have. Sometimes it matches the farthest destination but this is not necessarily always the case. The choice of $F_{\omega(i,j)}$ depends on the routing algorithm used. For instance, in a wNoC mesh with XY routing, as the one considered in this paper, the worst destination of flow $F_{\omega(i,j)}$ corresponds to the farthest node that can be reached from the next F_i hop's input port depending on the traversing direction.

The first question that raises from the contention formulation above is whether the assumptions on top of which this model is built are pessimistic or not. We want to know if considering that nodes in the network are injecting packets in an uncontrolled manner or, in other words, that the number of in-flight requests per node in the network is unlimited, is the reason why composable contention delay bounds are pessimistic [30].

Interestingly, as already shown in [30], limiting the injection of the flow under analysis, has no impact on $\overline{WCD}$ since this only affects intra-task contention and not the contention due to inter-task interferences. The reason is that time-composable WCET estimates require considering the worst possible interleaving of requests in the wNoC and this causes, in general, worst situations to be also possible when allowing only one in-flight request per flow. In Section 5 we corroborate this hypothesis with an empirical evaluation.

4 Probabilistic wNoC Designs

Unlike deterministic wNoCs, probabilistic network designs do not require the timing analysis to consider that all accesses systematically experience their worst possible contention. Instead, random events occur with true probabilities. Therefore, the probabilistic analysis made by MBPTA arises as a suitable approach to reduce the pessimism factored in the contention in wNoCs. To enable the derivation of pWCET estimates with MBPTA, two conditions must hold in the wNoC design: (i) conflicts in the wNoC must have a probabilistic nature (i.e. should occur with a given probability); and (ii) the execution conditions (contention) under which the timing measurements of the application are collected at *analysis* are actually an upper bound of those that will occur during *operation*. Condition (i) requires modifications in the arbitration unit of the router (Section 4.1) and condition (ii) requires defining a contention scenario which safely upper-bounds the worst possible one (Section 2.3). In this section we present how a COTS wNoC must be adapted to make them compatible with MBPTA and how to optimize them to make pWCET estimates tight, which are the main contributions of this work.

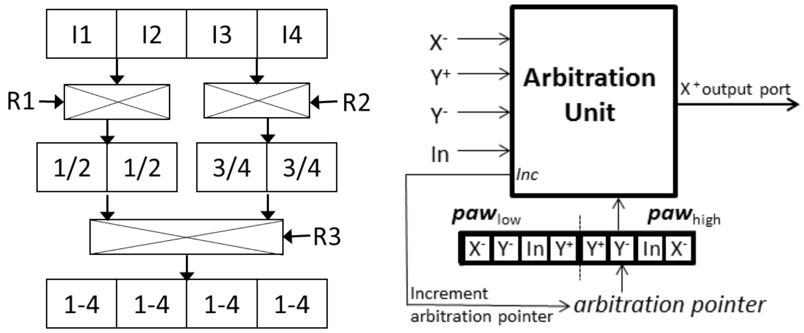
4.1 MBPTA-compliant wNoC Router Design

To make a wNoC design MBPTA-compliant, we have to make packet jitter follow a probabilistic behavior. To do so, hardware changes are required in the arbitration unit of the NoC router. An intuitive, but inefficient, MBPTA-compliant policy to grant access to a given output port is to simply select one of the requests randomly. This might cause a given request to take long (in theory infinite) time to be granted access. Instead, from the different MBPTA-friendly arbitration policies, we choose random permutations as it delivers superior performance and bounded contention [14]. Random permutations grant access to N contenders in a round-robin fashion, but in a random order. Such order changes every N arbitrations, so that each contender is granted access once every N slots, but in a random order.

To implement random permutations in the wNoC router, we modify the arbiter to be able to generate a random permutation P_i of all four inputs for every output port, where the four inputs and the output port belong to the group $(X+, X-, Y+, Y-, In)$. Whenever one or more packets request access to a given output port, the arbiter grants access according to P_i and an arbitration pointer. When a permutation is generated the arbitration pointer points to the first input port in the permutation. If the first input in the permutation is not requesting the output port then the next input port in the permutation is selected. This process is repeated until an input port with a pending request is selected. Then the arbitration pointer is moved to the input port in the permutation after the one granted access. When the pointer reaches the end of the permutation, a new random permutation window is generated.

Each output port has two probabilistic arbitration windows (*paw*), aka permutations with all inputs: the one being used (paw_{low}) and another one (paw_{high}). The generation of a new permutation occurs when the pointer reaches the end of paw_{low} so that the new permutation is available for the next arbitration. At that point, paw_{high} becomes paw_{low} and the new permutation becomes paw_{high}. This avoids generating an arbitration window at the same time that one of its elements needs to be selected. Figure 3 shows how the proposed random arbiter works. In particular, in the figure the arbitration pointer has been increased exceeding the boundaries of paw_{low}. At this time, paw_{high} would become paw_{low}, and paw_{low} would be initialized with a new random permutation.

Hardware implementation. Random permutations can efficiently be implemented using a configuration like the one shown in Figure 3a. We implement



(a) Random permutations schematic

(b) Router integration

Figure 3: Schematic of the random permutations implementation in the wNoC router architecture.

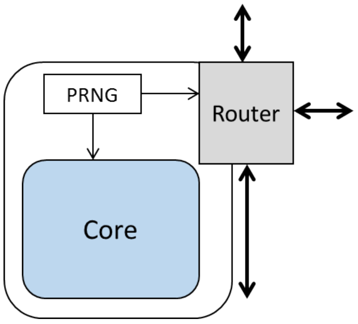


Figure 4: PRNG integration with the core and the router.

them for each of the 5 output ports using $N = 4$ inputs, so that they need only $N \cdot log_2 N$ bits for the register and $N - 1$ random bits. Those random bits are produced by a pseudo-random number generator (PRNG) as the one proposed in [6]. Additionally, the costs of implementing a PRNG generator can be minimized if the PRNG unit is shared between the core and the router as shown in Figure 4.

This router is the basic component on top of which the two probabilistic wNoC designs proposed later in this paper relies on. The random arbitration performed in this router allows leveraging the impact of contention in the measurements to probabilistically represent the contention in the wNoC. While arbitration decisions are random, they carry out dependences across arbiters since the actual requests contending for a given output port in a router often depend on the (random) decisions taken in other routers. Any state of the wNoC in terms of contention moves to any other state with a given probability due to the purely random nature of all arbitration choices. Therefore, each sequence of states occurring during the execution of the task under analysis occurs with a given probability and hence, each potential execution time has a true probability to occur, as needed to apply MBPTA.

4.2 Reducing Contention in Probabilistic wNoCs

By combining worst-contention scenarios with the probabilistic router architecture proposed in section 4.1 we can produce execution conditions during analysis that upper-bound those during operation. Execution time measurements collected under this setup can be used reliably to apply MBPTA in order to derive WCET estimates. However, if we do not anyhow limit the contention in the network, the stalls experienced by the requests of the task under analysis can be very high and thus, WCET estimates will account for high contention for all requests, similarly to the case of time-deterministic wNoCs. In time-deterministic wNoCs the worst-case contention with, for instance, round-robin arbitration, is accounted for all requests regardless of the degree of contention in the network. In the case of probabilistic wNoCs, requests experience the actual contention of the worst-case scenario modelled at analysis, which is enforced not to be exceeded during operation. Therefore, decreasing maximum contention by design opens the door to obtaining lower WCET estimates with probabilistic wNoCs, as already shown for tree wNoCs [39].

In order to decrease contention and derive tighter WCET estimates in the wNoC, we propose two mechanisms: (1) Limiting the number of in-flight requests or (2) limiting the injection frequency. The first approach is more suitable for applications that are sensitive to network latency while the latter for applications with high throughput requirements. It is important to mention that reducing contention by reducing the number of requests in the network is suitable for probabilistic wNoCs because, in such designs, requests interleave probabilistically and therefore, worst-case alignment of flows and arbitration decisions do not need to be accounted for systematically (as opposed to the case of time-deterministic wNoCs). In other words, already proposed techniques for reducing contention in time-deterministic wNoCs, such as injection throttling [41], cannot obtain tighter WCET estimates.

4.2.1 Limiting the number of in-flight requests (LNR)

Contention in the network can be reduced by limiting the number of requests in-flight for all the nodes in the network. With our proposed MBPTA-compliant router design, we remove the need to know the exact alignment and we only need to ensure that, during the analysis phase, the task under analysis can have *up to n* requests in-flight and all the other cores have *always* exactly n requests in flight. In this case, execution times obtained for the task under analysis are obtained under worst possible contention conditions.

Hardware support. LNR can be easily implemented at the network interfaces by having a counter for the number of requests in the network. This counter is increased when a new request is injected in the wNoC and it is decreased once a response from an injected request is received back at the node the task under analysis is attached to. Note that such an approach is conservative since an additional request could be sent once the target node has served it, but it is not sent until the response does not arrive back to the sender. If such hardware mechanism is in place, the generation of the maximum contention scenario at analysis can be implemented in software by using microbenchmarks that constantly send requests to the node that affects the most the flows of the core under analysis.

4.2.2 Limiting injection frequency (LFR)

In this case, to be able to derive tight contention delay bounds, we propose to control injection frequency at wNoC nodes. Similarly to the previous proposal, during the *analysis* phase the task is run in isolation. Requests of the task under analysis are allowed to be injected if at least Minimum Inter-request Delay (*MID*) cycles have elapsed since the previous request was injected. For all the other cores, we have to generate requests at the maximum allowed frequency, i.e. once every *MID* cycles to create maximum contention.

Hardware support. For the hardware implementation of LFR we need at the network interface a counter from 0 to up to MID-1 cycles to control the maximum frequency at which requests (from contenders and the core under analysis) are injected in the network. Stressing scenarios can be reproduced by software means as for LNR.

Since we do not assume any specific pattern and maximum contention is enforced during analysis, time-composability is preserved. During operation all cores can inject requests with the same restrictions imposed during the analysis (keeping up to n requests in-flight for LNR or at least *MID* cycles after the previous injection for LFR), which can only lead to equal or lower contention than that accounted for in the pWCET estimates.

5 Evaluation

5.1 Methodology

Target Processor Architecture. We model a wNoC-based manycore processor with pipelined in-order cores[2] with a simulator based on the SoCLib simulation framework [3]. Each core has separated first level instruction (I1) and write-through data (D1) caches, a partitioned-across-cores write-back L2 cache and main memory. I1 and D1 are 16KB, 4-way and 16B/line and the L2 has 128KB 4-way per core to discount L2 cache effects from the analysis. All caches implement random placement and replacement policies [16]. Hit/miss latencies are 1 and 3 cycles for I1/D1 and 2 and 7 cycles for L2.

[2] Note that more complex processor cores, although compatible with our proposals, are very hard to time-analyze and thus, not suitable for hard-real time applications.

wNoC. We model the wNoC with an enhanced version of the gNoCsim [2] simulator, that has been integrated with the SoCLib framework. Cores and memories are connected using a mesh network topology with XY routing. For a NxN mesh we index routers from R(0,0) to R(N-1,N-1). The shared L2 cache memory and a shared memory controller are connected at router R(N-1,N-1). The memory controller implements random permutation policy. Two virtual networks are used to split requests and responses. Routers are pipelined and consist of 4 stages: input buffer, routing, switch allocation, and crossbar traversal. In line with other works [30, 29] in all wNoC setups we use packetization to have single-flit packets only. The number of VCs is 1. Additional virtual channels would not provide higher guaranteed performance in our setup, as discussed in [30].

Having packets with more than 1 flit would penalize WCET significantly. This occurs because, to achieve reliable estimates, each contending packet has to be assumed to have the maximum allowed length, making the maximum waiting time to get access to a specific output port grow. In particular, given that read (write) requests use 1 flit (4 flits for writes) and responses 4 flits (1 flit for writes), using packets of up to 4 flits would increase contention for the 1-flit packets by 4x, since contender packets must have the largest size to preserve time composability. Therefore, each communication involving 5 flits (either 1+4 for reads or 4+1 for writes) would experience around 60% higher contention with 4-flit packets than with 1-flit packets since, assuming A arbitration rounds lost per packet, a n-flit packet wNoC would cause $n \cdot A \cdot numpackets$ contention per communication (where n is the maximum packet size and $numpackets$ the number of packets sent). Hence, for 1-flit packets we would have $1 \cdot A \cdot 5 = 5 \cdot A$ contention, whereas for 4-flit packets we would have $4 \cdot A \cdot 2 = 8 \cdot A$ contention, thus a ratio of 8 vs 5. Hence, we use 1-flit packets since they minimize WCET estimates as discussed in [30].

The impact of virtual channels depends on the allocation technique used. When virtual channels are allocated dynamically, with the aim of reducing instantaneous head of line blocking situations, they have a negative impact on the worst contention a given flow may suffer since, in the worst-case, the task under analysis will have to wait for all packets in all VCs of each input port. With a static virtual channel allocation, there are some cases for which a particular VC allocation would reduce the contention due to decreasing the maximum contention that some specific flows can cause in the task under analysis. Unfortunately, in the context of all-to-one communication, since all flows target the same destination, head of line blocking is not a consequence of inter-flow conflicts but just a consequence of the limited ejection capabilities of the network.

Authors in [30] showed that measuring inter-task interferences in a wNoC using the Worst Contention Delay ($\overline{WCD}$) metric results in much tighter WCET estimates than using Worst-Case Traversal Time (WCTT) and it allows obtaining time-composable WCET estimates. Therefore, we compare our probabilistic wNoC design with a deterministic setup that experiences $\overline{WCD}$. For synthetic traffic results, the flow under analysis experiences $\overline{WCD}$ since all flows in the wNoC are injecting packets at the maximum rate allowed. For obtaining WCET

Table 2: I.i.d. test results for LNR1 and LFR20 on a 4x4 mesh setup.

Benchmark	a2time	aifirf	basefp	bitmnp	cacheb	canrdr	iirflt	pntrch	puwmod	rspeed	tblook	ttsprk
LNR1 indep	0.47	0.20	0.74	0.34	0.41	0.48	0.86	0.24	0.98	0.63	0.99	0.48
LNR1 i.d.	0.31	0.99	0.53	0.78	0.49	0.06	0.60	0.16	0.14	0.51	0.12	0.79
LFR20 indep	0.47	0.07	0.82	0.53	0.30	0.25	0.44	0.88	0.17	0.30	0.47	0.35
LFR20 i.d.	0.34	1.00	0.26	0.42	0.70	0.39	0.42	0.71	0.41	0.53	0.24	0.15

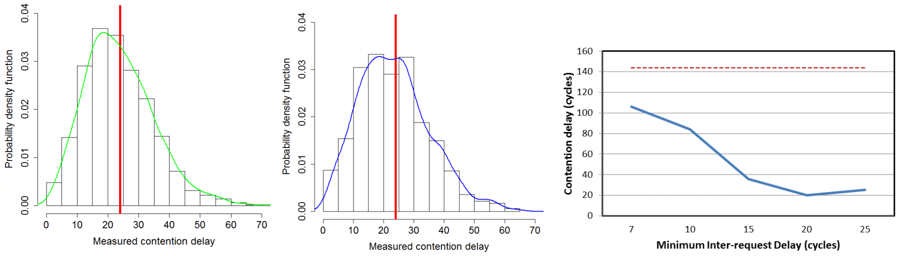
estimates, we artificially enforce requests to experience $\overline{WCD}$ to make simulation times affordable. The rest of the platform features are kept identical in both setups, deterministic and probabilistic ones, to understand the differences in the guaranteed performance provided by the wNoC since both setups are compatible with MBPTA.

Workload. For characterizing wNoC performance we use synthetic traffic. Worst-contention scenarios are created by artificially injecting requests in the wNoC targeting worst-possible destinations and at the maximum allowable rate. As representative real-time workloads we use the EEMBC Autobench suite [31], which reflects current real-world demand of some automotive critical real-time embedded systems. Hence, each benchmark is run in the core that theoretically can experience highest contention due to contending flows[3], and the other cores inject traffic sustainedly at the highest rate allowed targeting the node that leads to highest contention for the task under analysis[4]. This analysis scenario upper-bounds the contention that any real workload could produce, thus delivering time-composable pWCET estimates. During operation, contention due to other tasks cannot be higher than the one enforced during analysis (in fact it will be typically much lower), so analysis time measurements can be reliably used for pWCET estimation. Further details on probabilistic upper-bounding in NoCs can be found in Section 2.3.

WCET estimation. We follow MBPTA process to obtain pWCET estimates. For each task we collect 1000 runs and present pWCET estimates for a cutoff probability of 10^{-13} per run, although the same trends are obtained for other cutoff probabilities. As part of the MBPTA application process, execution time samples (of 1000 runs each) have been statistically tested for independence and identical distribution, a prerequisite for the use of EVT. In particular, we have used the Ljung-Box independence test and the Kolmogorov-Smirnov two-sample identical distribution tests as prescribed in [4], with a significance level $\alpha = 0.05$, thus meaning that i.i.d. hypothesis is rejected only if one of the tests delivers a result below 0.05. All execution time samples passed those tests. Table 2 provides results for **LNR** with 1 in-flight request (LNR1) and **LFR** with $MID = 20$ (LFR20) for both tests on a 4x4 mesh setup for illustration

[3] While determining the core that could suffer highest contention is done as in time-deterministic wNoCs, we have empirically verified that such core (the farthest one from destination) is the one leading to highest pWCET estimates by repeating the experiments placing the task under analysis in each core in the multicore.

[4] While we perform such injection with hardware means, we have verified empirically that creating tasks that perform sustained requests to the same target nodes produces exactly the same effect (probabilistically).



(a) 1 request 3x3 (LNR) (b) 5 requests 3x3 (LNR) (c) Contention in a 4x4 with LFR

Figure 5: Contention in 3x3 and 4x4 wNoC setups with LNR and LFR.

	3x3	4x4	6x6
LNR	0.333	0.119	0.118
LFR	0.856	0.795	0.327
DET	0.333	0.111	0.007

Table 3: Minimum guaranteed throughput w.r.t ideal case. DET refers to a deterministic setup.

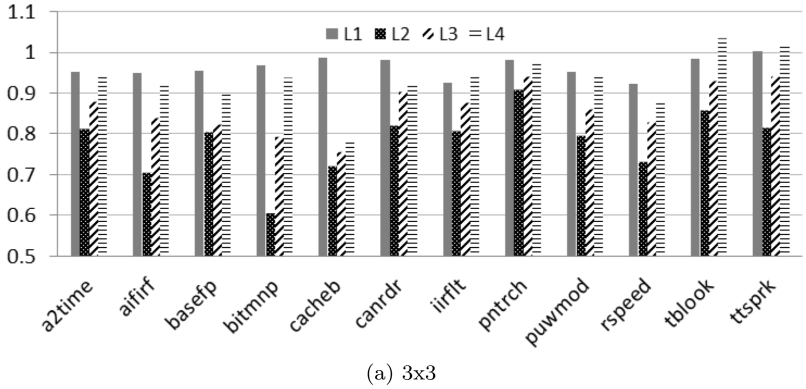
purposes. As explained, all tests deliver results above the significance level, so the i.i.d. hypothesis cannot be rejected.

5.2 Characterizing wNoCs Performance

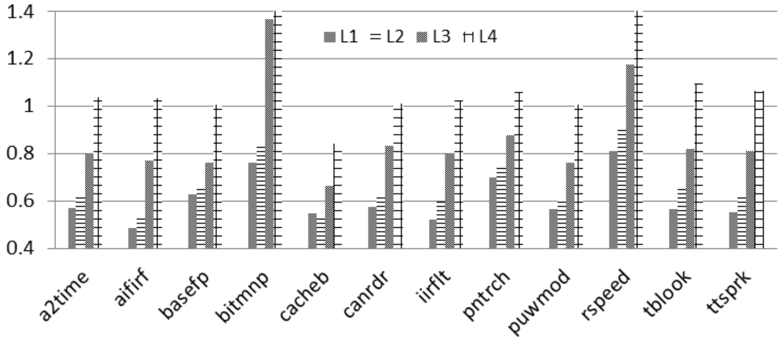
We have used synthetic traffic to characterize the performance of the probabilistic wNoC. Figure 5 shows the measured contention when nodes have 1 (a) and up to 5 (b) requests in a 3x3 network setup for the flow that goes from $R(0,0)$ to $R(2,2)$. The vertical red line represents the WCD for these flows and it has been computed analytically [30]. As shown, contention in a probabilistic wNoC setup follows a probabilistic distribution. This probability distribution is centered around $\overline{WCD}$ (the worst contention in a deterministic wNoC setup). Interestingly, the shape and location of the distribution is almost identical regardless of the number of in-flight requests, one or five.

The conclusions we draw from this analysis are twofold. First, as shown in [30], worst-contention situations in the wNoC are also possible when allowing only one request in-flight per core. Second, probabilistic wNoCs do not allow reducing network contention per se as in average the worst contention experienced in the network remains the same.

Figure 5c shows the results of an additional experiment that analyzes to what extent the contention in the network can be reduced by controlling the frequency at which network interfaces issue requests. In this plot, the analytically computed WCD is represented by the horizontal dashed red line. As shown, average contention delay for a 4x4 mesh decreases for our probabilis-



(a) 3x3



(b) 4x4

Figure 6: LNR pWCET estimates normalised w.r.t. a deterministic wNoC (Ln means up to n requests in-flight allowed).

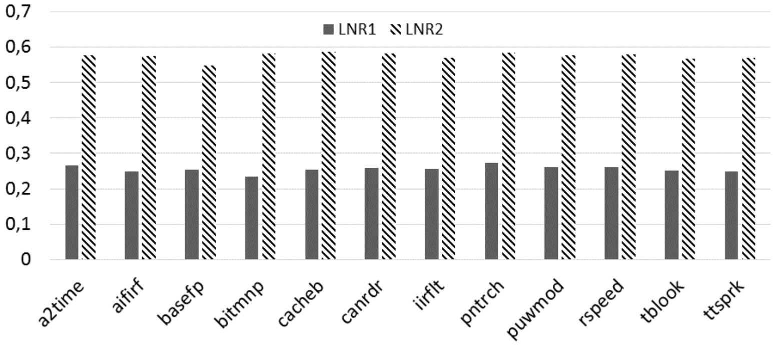


Figure 7: **LNR** for 6x6 mesh normalized w.r.t. a deterministic wNoC.

tic wNoC as we decrease injection frequency flattening when the total injection rate of the mesh leads to a non-saturated network so that traffic effects decrease. Contention starts to decrease at an injection frequency of 1 request every 16 cycles per core, so that all 16 cores inject at most 1 request per cycle in total on average, which is the maximum ejection rate of the network when all nodes target the same destination. On the other hand, with decreasing injection frequency, we also postpone requests of our core, so then we need to find a good balance between low contention and large delay for our requests. Furthermore, we also see how the WCD in the 4x4 mesh is much higher than in the 3x3 setup due to the non-linear increase in the number of flows contending in the wNoC.

Table 3 shows minimum per node guaranteed throughput normalised w.r.t. the ideal scenario in which a node in a NxN wNoC setup is able to eject flits from each of the nodes at a 1/(NxN) rate. As shown in the table, both probabilistic setups outperform guaranteed throughput values provided by the deterministic wNoC setup and LFR is the one able to retrieve more bandwidth. The deterministic wNoC setup provides very poor bandwidth guarantees for big and medium size wNoCs because of the distributed nature of the arbitration, which allocates a very small fraction of the bandwidth to the farthest nodes when the network is fully congested [29].

5.3 Performance Evaluation

For evaluating the performance guarantees of our probabilistic wNoC setups, we use the EEMBC single-threaded workloads as the task under analysis. In these experiments, the task under analysis is placed at the node attached to $R(0,0)$ and the rest of the cores are forced to cause worst-possible contention as described in Section 4. Figures 6 and 7 show the pWCET estimates achieved by limiting the number of in-flight requests (**LNR**) for the different benchmarks. Results are normalized w.r.t. the case of the deterministic wNoC. All the existing task communications target the shared memory controller that is attached to $R(2,2)$ in the 3x3 case and to $R(3,3)$ and $R(5,5)$ in the 4x4 and 6x6 case[5]. Other possible task placements provide similar comparative results. As shown in Figures 6 and 7, **LNR** wNoC setup outperforms the performance guarantees achieved by the deterministic wNoC design.

LNR trades off two opposite effects: a low number of in-flight requests decreases contention but delays request issue times, and vice versa. For small wNoCs (e.g. 3x3), the optimal point consists of allowing two in-flight requests per core since, despite the increased relative contention, the wNoC is able to eject requests fast enough due to the low number of cores. As we increase the core count (e.g. 4x4), contention becomes the dominant effect since more cores inject requests whereas the ejection rate remains as for smaller wNoCs (1 request per cycle). Hence, the number of in-flight requests must be kept as low as possible. Still for 4x4 wNoCs, the difference between 1 and 2 in-flight requests is small. However, such difference grows dramatically when using larger wNoCs

[5] Although our approaches scale smoothly regardless the core count, we do not consider larger manycores due to the increasingly poor scaling of deterministic wNoCs for larger core counts.

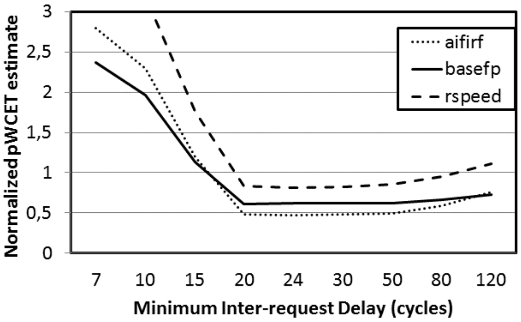


Figure 8: LFR pWCET estimates w.r.t a deterministic wNoC in a 4x4 mesh.

(e.g. 6x6 in Figure 7), and allowing only 1 in-flight request is, by far, the best choice.

It might not seem obvious the reason why **LNR** behaves better than a deterministic setup since, as already shown in Figure 5, requests in both setups experience the same average contention. The reason lies in the way contention is distributed. For a deterministic approach, where a particular alignment of requests cannot be assumed, it needs to be considered systematically that requests will be absorbed at a constant rate that is equal to $\overline{WCD}$. On the contrary, in a probabilistic approach roughly 50% of the requests will get absorbed faster than $\overline{WCD}$. These fast requests make it possible to take advantage of the store buffer of the pipeline more frequently than for a deterministic approach and allow a higher overlapping between computation and communication, thus leading to smaller execution time[6]. In particular, the behavior of the deterministic wNoC is a *fill-and-stall* behavior of the store buffer in front of store bursts, thus stopping pipeline progress always. Conversely, the probabilistic wNoC allows releasing store buffer entries sometimes earlier due to lower contention delay, and for the time a new store arrives at the store buffer, there is space available so that the pipeline keeps progressing in parallel with the processing of stores in the wNoC.

Figures 8 shows pWCET estimates for the **LFR** approach for 4x4 network setups. Note that the plot shows only results for three benchmarks representing the best, the average and the worst case. As we can see, the improvement achieved by the **LFR** approach is very significant being 40% on average for the case of 20 Minimum Inter-request Delay cycles. In general, **LFR** reaches the best results when the injection frequency (MID) is the highest (so lower number of cycles between request injections), but without exceeding the ejection rate, which is 1 request per cycle. Hence, given a wNoC with N cores, MID must be strictly above N cycles not to saturate the wNoC, which would lead a fully-congested wNoC, thus with very high execution times and pWCET estimates. For instance, in the case of the 4x4 setup, $MID \geq 16$. In general, if the overall injection rate of the wNoC is close to the ejection rate (e.g. $MID \rightarrow 16$ for

[6] This holds for timing-anomaly-free processors like the one used in our experiments.

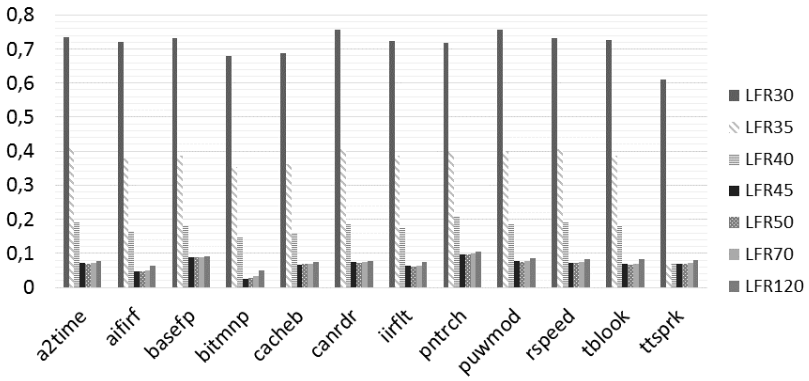


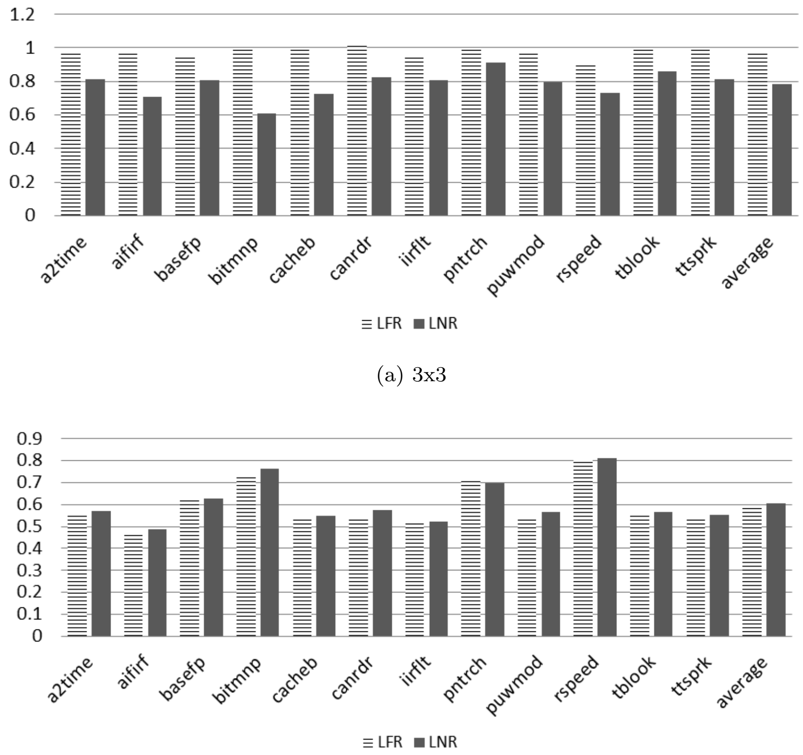
Figure 9: **LFR** for 6x6 mesh normalized w.r.t. a deterministic wNoC (LFRn stands for LFR with *MID*=n).

a 4x4 wNoC), there is still some meaningful contention, so the sweet spot is typically a *MID* value a bit higher than the wNoC core count.

If we move to a 6x6 mesh, **LFR** brings huge guaranteed performance benefits: in Figure 9 we see that we have 93.3% improvement on average for $LFR50$ (i.e. **LFR** with *MID*=50 cycles). This occurs because latency bounds grow linearly with distance with **LFR**, whereas they grow exponentially for time-deterministic approaches, that need to account for the worst potential behavior for all requests, which is extremely unlikely and far above the typical case for **LFR**. If we compare **LNR** and **LFR** for the 6x6 setup, we observe that **LNR1** provides pWCET estimates 75% lower than those on a deterministic network, but this is still almost 4x higher than **LFR**, which is clearly the best choice as the size of the mesh grows.

Figure 10 shows how LFR and LNR aproaches outperform significantly the results of the deterministic wNoC. We only show 3x3 and 4x4 results since deterministic bounds for the 6x6 network are too high to make it usable (more than 5,000 cycles for the farthest flow). A direct comparison of results for both approaches, **LNR** and **LFR**, for 3x3, 4x4 and 6x6 network setups is shown in Figure 11. In this plot, we build on the results for the the best possible configuration for **LNR** and **LFR**. Note that the best configuration, the number of in-flight requests in the case of **LNR** and the actual frequency of requests for the case of **LFR** providing the best performance, only depends on the properties of the task under analysis and not on the actual load that co-runners put in the network.

Finally, we have measured how far **LNR** and **LFR** pWCET estimates are from the ideal case. To do so, we compare the results of probabilistic wNoC setups with the results for tasks running in isolation, so experiencing always zero-load delay. On the one hand, for small networks **LNR** achieves pWCET results that are relatively close to the case without any contention, on average 9% for the 3x3 setup and 34% for 4x4, whereas for 6x6 it is around 330% above



(a) 3x3

(b) 4x4

Figure 10: **LNR** and **LFR** WCET estimates normalised to the WCET of the deterministic setup.

the no-contention scenario. On the other hand, when increasing the size of the network the performance cost of the **LFR** mechanism drops significantly: 35% and 32% for 3x3 and 4x4 setups respectively, and only 5% for the 6x6 setup w.r.t. no contention since requests are much less likely to contend anywhere in the NoC when the NoC size increases. Therefore, **LNR** and **LFR** are complementary solutions fitting small and large NoCs respectively.

5.4 Implementation Overheads

The proposed wNoC designs require the introduction of some modifications in the router and the network interfaces. In this section, we evaluate the area requirements of the proposed solution w.r.t. canonical wNoC router requirements. Area estimates are based on synthesis results using the 45nm nangate library [1]. For the canonical router implementation, we used the architecture described in [34]. The area of this canonical 2D-mesh routing with XY routing implementation and 4 flit buffers is $17651\mu m^2$ with no virtual channels and

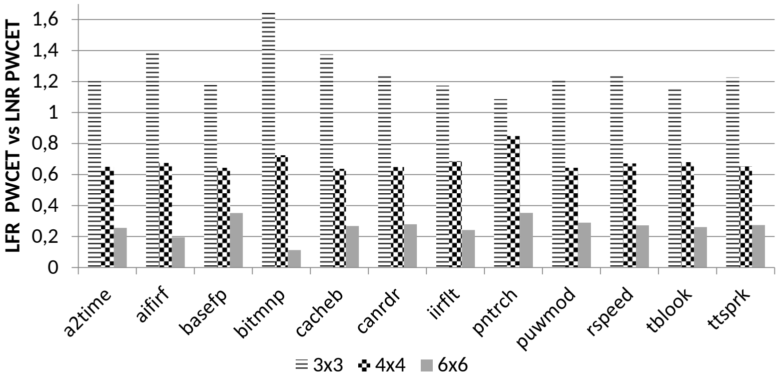


Figure 11: Scalability of the LNR and LFR approaches. Results show LNR pWCET divided by LFR pWCET.

$89976 \mu m^2$ when implementing 5 virtual channels.

Randomizing the arbitration requires producing some random bits using a PRNG. For the PRNG, we opt for the implementation proposed in [6] that has been shown compatible with MBPTA requirements. This implementation requires a 168-bit linear feedback shift register producing 32 random bits. This PRNG implementation requires $691.2 \mu m^2$. However, since MBPTA also requires cores implementing randomization features the overhead of producing these random bits for the router arbitration is minimized. In fact, out of these 32 bits, only 15 bits are required by the wNoC router proposed and the remaining 17 bits are sufficient to randomize the required core features as described in [6].

Random permutation arbitration can be implemented as described in Section 4. Following this approach, the random permutation implementation in a 2D mesh router requires $224 \mu m^2$ for arbitrating the access to the 5 output ports.

In summary, the proposed router modifications are affordable involving area overheads equivalent to 3.1% of the area of a canonical wNoC router without virtual channels and 0.6% of the area of a 5 virtual channel router. These overheads include both the arbitration logic for the random permutations and the area corresponding to 15-bits of the PRNG.

5.5 Limitations and Future Extensions

As discussed in this section, we target systems with single-threaded tasks, thus meaning that no cache coherence support is needed in the hardware. In particular, data-sharing is restricted to specific OS routines and communication between tasks/partitions is performed through memory, as suggested by AR-

INC653 [5]. Second, cache coherence protocols have not been yet shown time-analyzable, so their use in the context of real-time is generally precluded. Overall, tasks do not share any data within time partitions and communication across tasks occurs at the boundaries of those time partitions, when the OS copies data from the output buffer of the producer to the input buffer of the consumer. Then, the OS flushes all caches at time partition boundaries, thus preserving cache coherence without any specific hardware support.

By considering single-threaded applications, and for the sake of facilitating timing analysis, cache space is either private per core (e.g. DL1) or partitioned (e.g. shared L2 cache). These configurations avoid interference in shared cache contents since other cores are neither allowed to alter the private nor the shared (partitioned) cache contents of any given core. Hence, this allows estimating the WCET (the pWCET in our case) of any given task *in isolation* as long as contention conditions considered in the access to shared resources (e.g. the wNoC) match or upper-bound those during operation.

Overall, our approach is proven compatible with systems building upon partitioned shared caches and software-managed cache coherence. However, on the other hand, our approach is not necessarily restricted to these configurations and could be extended to systems allowing multi-threaded tasks and/or non-partitioned shared caches. The use of our approach would require the use of measurement collection protocols where conditions observed during analysis match or upper-bound probabilistically those occurring during operation in all relevant aspects including (1) contention in the access to shared resources, (2) interference in shared cache space, and (3) interference due to coherence both in terms of wNoC messages as well as cache interference. While cache space interference has already been shown to be compatible with MBPTA [38], how to integrate those designs with wNoCs and multi-threaded applications remains as future work.

6 Related work

We classify the existing research in NoCs for real-time applications in the following four categories: (1) Real-time specific NoCs, (2) NoC calculus, (3) analytical worst-case bounds, and (4) probabilistically analyzable interconnects. Our work fits in the 4^{th} category and represents the first general realization of wNoC designs for probabilistic time analysis. Main differences across the four approaches are summarized in Table 4.

Real-time specific NoCs. We distinguish two main groups in this category. (1a) NoCs using TDMA [12] allow an easy derivation of composable WCET estimates but require specific designs only suitable for real-time applications that provide poor average performance. (1b) Approaches using flit-level virtual-channel (VC) prioritization [36] bound the contention in the NoC by forwarding first flows with higher priority. In general, flit-level VC prioritization schemes require abundant VC resources, although some approaches have shown how the amount of VCs can be effectively reduced in certain scenarios [26, 27].

	Bounds Computation	Composable WCET	HW Changes	Performance	Real-time
(1a)	Analytical	Yes	Deep	Low	Hard
(1b)	Analytical	(*)Partial	Moderate	Moderate	Hard
(2a)	NoC Calculus	No	None	High	Hard
(2b)	NoC Calculus	No	None	Very High	Soft
(3)	Analytical	Yes	None	Low	Hard
(4)	Not needed	Yes	Moderate	Moderate	Hard

Table 4: Approaches for achieving NoC performance guarantees. Category (4) includes the wNoC proposed. (*)Composability of this approach is restricted to the flows using the highest priority.

VC prioritization schemes require knowing the exact details of the critical flows in the network to derive WCET estimates, which complicates achieving time-composable estimates.

Network Calculus. Works based on Network Calculus [21] abstract communication flows using arrival curves that upper-bound the amount of traffic within any time interval. With the upper-bounded arrival curve and lower-bounded service curve, delay bounds can be derived. Time composability is lost with this approach unless worst possible traffic conditions are considered and (2a) deterministic – rather than (2b) stochastic [23] – delay bounds are used [32]. However, these assumptions defeat the whole purpose of Network Calculus. Thus, Network Calculus is appropriate when traffic information available during analysis is accurate, which may be for off-chip traffic, but is generally unaffordable for on-chip traffic.

Analytical wNoCs bounds. Another set of works focuses on determining wNoC packets worst-case traversal time (WCTT) by (3) considering worst-case conditions, first assuming limitations on the packet-injection rate [22], and later without this limitation [33, 11]. Finally, authors in [30] showed that measuring inter-task interferences in a wNoC using the Worst Contention Delay ($\overline{WCD}$) metric results in much tighter WCET estimates than using WCTT. The feasibility of achieving composable estimates in the context of time-deterministic wNoCs has been analysed in [30] and [42].

MBPTA compliant interconnects. (4) MBPTA compliance has been achieved for bus designs [14] either using Lottery arbitration [19] or proposing random permutations. TDMA-based buses have also been proven amenable for MBPTA by padding execution time measurements conveniently [28]. A tree NoC implementing wormhole routers with random arbitration and intended for all-to-one communication has also been shown to be amenable for MBPTA [39]. However, trees do not fit well all-to-all communication.

Summary. Our work targets achieving time-composable WCET estimates on high-performance wNoC designs for all-to-all communication. To do so, we rely on existing MBPTA randomization techniques. Like [23], [7] we exploit probabilistic analysis to avoid overdimensioning network contention. However, we introduce modifications (randomization) in the network that make contention to have by construction a probabilistic behavior instead of modeling application traffic probabilistically. Thus, we enable the derivation of WCET estimates with MBPTA by smartly limiting contention.

7 Conclusions

In this paper we show that appropriate probabilistic approaches are highly efficient dealing with contention in wNoCs. Pathological worst-contention scenarios occur with (provable) negligible probability and hence, there is no need to account for them. We propose two different wNoC setups, **LNR** and **LFR**, that are able to provide much better performance guarantees than deterministic approaches by making use of a wormhole router with randomized arbitration. **LNR** is particularly suitable for scenarios with moderate $\overline{WCD}$ values and for applications that are very sensitive to latency, and **LFR** suits better large NoCs where $\overline{WCD}$ values are expected to be huge.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the PROXIMA Project (www.proxima-project.eu), grant agreement no 611085. This work has also been partially supported by the Spanish Ministry of Science and Innovation under grant TIN2015-65316-P and the HiPEAC Network of Excellence. Mladen Slijepcevic is funded by the *Obra Social Fundación la Caixa* under grant Doctorado "la Caixa" - Severo Ochoa. Carles Hernández is jointly funded by the Spanish Ministry of Economy and Competitiveness (MINECO) and FEDER funds through grant TIN2014-60404-JIN. Jaume Abella has been partially supported by the MINECO under Ramon y Cajal postdoctoral fellowship number RYC-2013-14717.

References

[1] *The NanGate 45nm Open Cell Library. http://www.nangate.com.*

[2] *NanoC: NaNoC design platform. http://www.nanoc-project.eu.*

[3] Soclib, http://www.soclib.fr/trac/dev, 2012.

[4] Jaume Abella, Maria Padilla, Joan Del Castillo, and Francisco J. Cazorla. Measurement-based worst-case execution time estimation using the coefficient of variation. *ACM Trans. Des. Autom. Electron. Syst.*, 22(4):72:1–72:29, June 2017.

[5] Aeronautical Radio, Inc. *Avionics Application Software Standard Interface: ARINC Specification 653.* Aeronautical Radio, Inc, 2013.

[6] I. Agirre et al. IEC-61508 SIL3 compliant pseudo-random number generators for probabilistic timing analysis. In *DSD*, 2015.

[7] P. Bogdan, M. Kas, R. Marculescu, and O. Mutlu. Quale: A quantum-leap inspired model for non-stationary analysis of noc traffic in chip multi-processors. In *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, pages 241–248, May 2010.

[8] F.J. Cazorla et al. Upper-bounding program execution time with extreme value theory. In *WCET Workshop*, 2013.

[9] L. Cucu-Grosjean et al. Measurement-based probabilistic timing analysis for multi-path programs. In *ECRTS*, 2012.

[10] Sujay Deb, Kevin Chang, Xinmin Yu, Suman Prasad Sah, Miralem Cosic, Amlan Ganguly, Partha Pratim Pande, Benjamin Belzer, and Deukhyoun Heo. Design of an energy-efficient cmos-compatible noc architecture with millimeter-wave wireless interconnects. *IEEE Transactions on Computers*, 62(12):2382–2396, 2013.

[11] T. Ferrandiz et al. A sensitivity analysis of two worst-case delay computation methods for spacewire networks. In *ECRTS*, 2012.

[12] K. Goossens, et. al. Aethereal network on chip: concepts, architectures, and implementations. *Design Test of Computers, IEEE*, 2005.

[13] International Organization for Standardization. *ISO/DIS 26262. Road Vehicles – Functional Safety*, 2009.

[14] J. Jalle et al. Bus designs for time-probabilistic multicore processors. In *DATE*, 2014.

[15] B. Kim et al. A real-time communication method for wormhole switching networks. In *ICPP*, 1998.

[16] L. Kosmidis et al. A cache design for probabilistically analysable real-time systems. In *DATE*, 2013.

[17] L. Kosmidis et al. Fitting processor architectures for measurement-based probabilistic timing analysis. *Microprocess. Microsyst.*, 47(PB):287–302, November 2016.

[18] S. Kotz and S. Nadarajah. *Extreme value distributions: theory and applications*. World Scientific, 2000.

[19] K. Lahiri et al. Lotterybus: a new high-performance communication architecture for system-on-chip designs. In *DAC*, 2001.

[20] S. Law and I. Bate. Achieving appropriate test coverage for reliable measurement-based timing analysis. In *ECRTS*, 2016.

[21] J.-Y. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet.* Springer-Verlag, 2001.

[22] Sunggu Lee. Real-time wormhole channels. *Journal Of Parallel And Distributed Computing*, 63:299–311, 2003.

[23] Z. Lu et al. Towards stochastic delay bound analysis for network-on-chip. In *NoCS*, pages 64–71, 2014.

[24] E. Mezzetti and T. Vardanega. On the industrial fitness of wcet analysis. In *WCET Workshop*, 2011.

[25] M. Di Natale, J. Abella, J. Reineke, A. Hamann, and G. Farrall. Predictable system timing – probab(ilistical)ly? In *DAC (panel in automotive track)*, 2016.

[26] Borislav Nikolic, Hazem Ismail Ali, Stefan M. Petters, and Luís Miguel Pinho. Are virtual channels the bottleneck of priority-aware wormhole-switched noc-based many-cores? In *21st International Conference on Real-Time Networks and Systems, RTNS 2013, Sophia Antipolis, France, October 17-18, 2013*, pages 13–22, 2013.

[27] Borislav Nikolic and Luís Miguel Pinho. Optimal minimal routing and priority assignment for priority-preemptive real-time nocs. *Real-Time Systems*, 53(4):578–612, 2017.

[28] M. Panic, et. al. Enabling TDMA arbitration in the context of MBPTA. *DSD*, 2015.

[29] M. Panic, et. al. Improving performance guarantees in wormhole mesh noc designs. In *DATE*, 2016.

[30] M. Panic, et. al. Modeling high-performance wormhole nocs for critical real-time embedded systems. *RTAS*, 2016.

[31] J.A. Poovey et al. A benchmark characterization of the EEMBC benchmark suite. *IEEE Micro*, 29, 2009.

[32] Y. Qian et al. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In *NoCS*, pages 44–53, 2009.

[33] D. Rahmati, et. al. Computing accurate performance bounds for best effort networks-on-chip. *IEEE Transactions on Computers*, 2013.

[34] Antoni Roca. *Floorplan-Aware High Performance NoC Design.* PhD thesis, Universitat Politecnica de Valencia, 2012.

[35] A. Shacham, K. Bergman, and L. P. Carloni. Photonic networks-on-chip for future generations of chip multiprocessors. *IEEE Transactions on Computers*, 57(9):1246–1260, Sept 2008.

[36] Z. Shi and A. Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *NoCS*, 2008.

[37] Z. Shi and A. Burns. Real-time communication analysis with a priority share policy in on-chip networks. *ECRTS*, 2009.

[38] M. Slijepcevic et al. Time-analysable non-partitioned shared caches for real-time multicore systems. In *DAC*, 2014.

[39] M. Slijepcevic, et. al. pTNoC: Probabilistically time-analyzable tree-based noc for mixed-criticality systems. In *DSD*, 2016.

[40] Z. Stephenson, J. Abella, and T. Vardanega. Supporting industrial use of probabilistic timing analysis with explicit argumentation. In *INDIN*, 2013.

[41] Mithuna Thottethodi, Alvin R. Lebeck, and Shubhendu S. Mukherjee. Self-Tuned Congestion Control for Multiprocessor Networks. In *HPCA*, 2001.

[42] S. Tobuschat and R. Ernst. Real-time communication analysis for networks-on-chip with backpressure. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 590–595, March 2017.

[43] F. Wartel et al. Timing analysis of an avionics case study on complex hardware/software platforms. In *DATE*, 2015.