ECRTS 2016

5-8 July
Toulouse, France

Work-in-Progress Proceedings

July 6th, 2016
Toulouse, France

Edited by Sebastian Altmeyer

Message from the Session Chair

It is my pleasure to welcome you to the Work-in-Progress (WiP) session of the 28^{th} Euromicro Conference on Real-Time Systems (ECRTS 2016). The ECRTS'16 WiP session is devoted to new and on-going research on real-time systems and applications. Its primary purpose is to provide researchers with an opportunity to discuss their evolving ideas and gather feedback from the real-time community at large.

The WiP session 2016 features eight excellent WiP papers covering a wide spectrum of real-time topics. I am confident that many of the research contributions we feature here will appear as full-fledged conference and journal papers in the near future. The proceedings are published online on the ECRTS 2016 WiP website:

`http://ecrts.eit.uni-kl.de/wip16`.

The presentations in this session are only intended to provide a brief overview of the ideas and approaches. The authors of each paper have also prepared a poster and are happy to discuss their work during the reception immediately following the presentations.

I would like to take the opportunity to express my gratitude to the members of the technical program committee for thoroughly reviewing all papers within a very short time frame. Furthermore, I would like to thank the organizers of ECRTS 2016, Christian Fraboul, Nathan Fisher, Jérôme Ermont, Jean-Luc Scharbarg, Rob Davis and Gerhard Fohler for their support in organizing the WiP session.

On behalf of the program committee, I wish you a pleasant Work-in-Progress session, hope you will enjoy the presentations and invite you to discuss the presented ideas with the authors during the reception.

Sebastian Altmeyer
University of Amsterdam
Session Chair

Technical Program Committee

Leandro Soares Indrusiak, University of York, UK
Martina Maggio, Lund University, Sweden
Linh Thi Xuan Phan, University of Pennsylvania, USA
Jan Reineke, Saarland University, Germany
Wolfgang Puffitsch, Technical University of Denmark, Denmark

Session Chair

Sebastian Altmeyer, University of Amsterdam, The Netherlands

Table of Contents

Enabling predictable parallelism in single-GPU systems with persistent CUDA threads

Paolo Burgio
University of Modena and Reggio Emilia, Modena, Italy
paolo.burgio@unimore.it

***Abstract*—Graphics Processing Unit, or GPUs, have been successfully adopted both for graphic computation in 3D applications, and for general purpose application (GP-GPUs), thank to their tremendous performance-per-watt. Recently, there is a big interest in adopting them also within automotive and avionic industrial settings, imposing for the first time real-time constraints on the design of such devices. Unfortunately, it is extremely hard to extract timing guarantees from modern GPU designs, and current approaches rely on a model where the GPU is treated as a unique monolithic execution device. Unlike state-of-the-art of research, we try to "open the box" of modern GPU architectures, providing a clean way to exploit intra-GPU predictable execution.**

I. Introduction

In the last decade, increasing demand for low-energy computational power from the embedded world met the tremendous performance-per-watt potential of modern the Graphic-Processing Units (GPUs), opening the doors to the adoption these devices in the new generation of embedded systems. The NVIDIA Tegra family [1] is an example of a GPU-based System-on-Chip (SoC) explicitly designed for smartphones and tablets. Recently, there is an increasing interest to adopt GPUs also in automotive and avionics settings, imposing for the first time hard real-time constraints in their design. Unfortunately, GPUs are not tailored to real-time systems, due to the complex hardware structure. Their aggressively parallel designs extract the maximum performance from the hardware, but at the same time they also hassle the analyzability of the overall platform. As an addition, the non-openness of most GPU low-level drivers and firmware makes it cumbersome to treat them other than as a "monolithic" piece of hardware, which cannot be managed and exploited in a more flexible manner. A significant example is the *warp scheduler* of NVIDIA GPUs, whose mixed hardware/software structure, firmware and OS drivers represent the key added value of the provider hence they are not disclosed nor well documented. Although perfectly comprehensible from a business/market point of view, this decision prevents most of the academics world to do research on GPUs, especially in the field of real-time systems. Indeed, in the real-time domain, a deep comprehension of the device hardware architecture is paramount to achieve predictability/real-time guarantees also at the software level.

Some research has been carried on for supporting timing-accurate and predictable computing on GPUs: for instance, GPUSync [2] provides i) pinning mechanism ii) budgeting and iii) integration support for multi-GPU systems. However, all of the current approaches target multiple-GPU systems, where the device itself is treated as a unique "atomic" execution device. This paper described our ongoing effort to "open the box" of GPU devices (here specifically of NVIDIA devices, as a testbed), to explore whether it is possible to **extract predictability guarantees within a single GPU**. To the best of our knowledge, we are the first doing so, and we will treat the problem **exclusively from a software point of view**, that is, we will not propose any modifications to existing hardware.

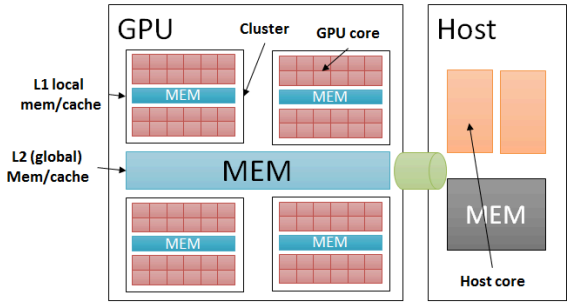


Fig. 1. Clustered GPU architecture

We believe that the generic clustered structure of a GPU device (shown in Figure 1) lends itself pretty well to being "opened" and treated at a finer-grained architectural level. We adopt a recent programming paradigm for GPUs which addresses a lightweight and flexible execution model employing *persistent GPU threads* [3], [4]. Persistent Threads run at user-level on the GPU, and contrarily to "traditional" GPU threads, they are not tailored for a specific computation, but continuously spin-wait for work to execute. As soon as the host subsystem (e.g., in consumer GPU systems, a multi-core CPU) wants to offload some work to the GPU, it sends to the persistent thread both a descriptor of the work and a reference to the in/out data items, which indicates the actual work to perform. This scheme can be repeated indefinitely. With such an approach, it is therefore possible to allocate work on a specific subset of GPU cores in such a way to minimize inter-cores interference (e.g., thrashing of global cache lines) and increasing overall platform predictability. We are currently developing an implementation named **LightKernel (LK)**, which we will release as open source software[1] to support research in this direction.

II. Approach and implementation of LightKernel

A. Approach and design choices

Before introducing the design of our LK architecture, it is important to point out a few choices that guided our implementation.

- We target real-time systems, where *predictability* is a primary concern. In such systems, it is perfectly normal to

[1]Feel free to download this preliminary implementation from: http://hipert.mat.unimore.it/LightKer

trade **average** performance for **worst case** performance, and to follow clean, modular software designs, which ensure spatial and timing isolation among application components.

- A typical optimization for GPU devices is to share the program counter (PC) register among groups of hardware cores. This causes the so-called *lockstep* execution, which means that cores belonging to the same group are tied to execute the very same assembly instruction, hence their execution cannot diverge. This mechanism is extremely useful when implementing data-parallel execution pattern.
- In modern GPUs, computing cores are physically partitioned into clusters (see Figure 1), with local (L1) caches and software-managed memory banks to maximize data locality to computation. More "lockstep groups" can belong to the same cluster. In such a system, exploiting local memories to maximize the locality of data to computing elements is a must to achieve performance. For this reason, we decide to expose clusters at the application level, providing low-level software subroutines to map both data and work on a specific cluster.

B. NVIDIA terminology and CUDA

Our implementation is based on CUDA [5], which is NVIDIA's proprietary API for programming GPUs. We chose these devices as a testbed because NVIDIA is undoubtedly the GPU market leader, and CUDA is one of the most widely adopted programming model for GPUs. Our approach can be seamlessly ported, e.g., to OpenCL [6] with minimal effort. In the NVIDIA terminology, the GPU device is composed of *CUDA cores*, clustered onto *Streaming Multiprocessors – SMs*. CUDA programmers partition the application onto a two-dimensional work space composed of *CUDA threads*, grouped onto *thread blocks*. The portion(s of application that are offloaded to the GPU device are called *CUDA kernels*. Within a kernel, typically, threads are mapped onto CUDA cores, while thread blocks are mapped onto GPU SMs. Lockstep islands are called *warps*, composed by multiple threads executing in "Single Program Multiple Data" – SPMD fashion. Intuitively, one thread block is composed of multiple warps, whose dimension is fixed for a given GPU architecture, and which are directly managed by the CUDA runtime. From the real-time point of view, warps are the data-parallel, non-preemptable, atomic unit of work which is schedulable on the GPU device, and programmers have no visibility nor control over them. We believe this is the biggest limitation of GPU architectures, from a real-time point of view.

C. LK persistent kernel

In a first step we follow the simplest design possible, and decide to statically *pin* persistent threads on GPU clusters at boot time, and to dynamically allocate work to a specific thread, that is, to a specific cluster of cores. This potentially enables fine-grained execution models, exploiting intra-GPU parallelism in a flexible way. This is possible because, as explained, "hardware lockstep islands" are confined within the single clusters. Roughly speaking, we spawn a single (persistent) CUDA kernel, made of B blocks of T CUDA threads, where B is the number of SMs in the target device, and T is the number of CUDA cores within a single SM, which is fixed for a given GPU architecture. What typically happens in GPU runtimes is that, when a CUDA kernel is spawned, its blocks are assigned to SMs in a round-robin fashion, so each LK block is mapped onto a dedicated SMs. We implemented a low-level checking mechanism for this (each thread within a block reads the SM number and compares it with the block ID which is assigned by CUDA runtime), which however was never triggered during our development and experiments.

Even if in modern GPUs it is possible to create more CUDA blocks (resp. threads) than available SMs (resp. cores), we want our design to be simple, and we don't spawn more threads than cores, and map exactly one CUDA block onto one SM [2]. We will explore more complex execution models, based on multiple persistent threads, in a future step.

Persistent thrd status	Value	Persistent thrd status	Value
from_GPU		*to_GPU*	
THREAD_INIT	0	THREAD_NOP	4
THREAD_FINISHED	1	THREAD_EXIT	8
THREAD_WORKING	2	THREAD_WORK	16+
THREAD_NOP	4		

TABLE I
PERSISTENT THREAD STATUSES AND MAILBOX VALUES

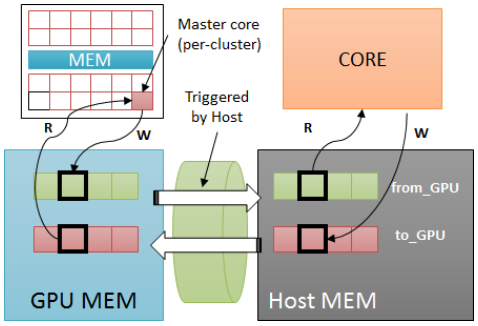


Fig. 2. Dual mailbox structure

D. Host-to-device communication

A key aspect of GPU computing is is the synchronization between the host and the device. We want to deliver work at the granularity of the single SM, and to do so, we implemented a dual lock-free mailbox system (see for instance [7]). In such a mechanism, each SM/persistent thread has two dedicated mailbox items (called `from_GPU` and `to_GPU`), as shown in Figure 2. We promote one CUDA thread within each block to act as *master thread* and to read-write the corresponding two mailboxes. Mailboxes are implemented as C `int`egers, whose values represent the statuses of LK-to-host communication protocol, as in Table I. In the considered architecture (see Section III) there are 16 SMs/clusters, hence our mailbox is composed by two arrays of 4x16 = 64 bits. Typically, GPUs are connected to CPU hosts via PCI EXPRESS connectors, delivering more than 15 GByte/sec. However, this is **peak** performance, i.e., it is delivered only with big data chunks, while the size of our mailbox is too narrow to efficiently exploit the mechanism, and we experience performance degradation in some cases. Unfortunately, data transfer from and to the device is mastered by OS driver for the specific device, which

[2]Architectural features such as, e.g., number of CUDA cores and SMs in a given GPUs can be easily obtained using platform-specific hooks.

is proprietary and undisclosed, so there is not much chance to improve this mechanism without accessing its internals.

III. Preliminary evaluation campaign

We performed a set of experiments to compare our LK to "standard" CUDA kernels. We expect the mailbox mechanism to be much lighter of traditional kernel spawn, with a significant performance gain. The only point of concern, as introduced in Section II, is the efficiency of transmitting few data chunks (i.e., the mailbox) across the PCI connector. We performed the experiments on a machine with the state-of-the-art of consumer GPUs by NVIDIA, the GTX980 mode, with the host running Ubuntu 14 Linux on an Intel i7 quad-core with 8GB RAM clocked at 3.6 GHz.

To structure the experiments, we split LK execution in a *Init* phase, to boot the system, and a *Dispose* phase, to release the GPU resource. In the middle, following the persistent thread model, we perform multiple subsequent stages to offload work to the device. Each of this stages is composed of four phases, namely *Copyin* of data in the device memory, *Trigger* of one or multiple SMs, *Wait* for one or multiple SM, and *Copyout* of data to the host memory. Similarly, traditional CUDA kernels, are made of an *Alloc* phase, to initialize the device driver and data buffers, *Copyin*, *Launch*, *Wait*, *Copyout*, and *Dispose* phase. We currently will not focus on data transfer effect, i.e., we do not consider the Copyin and Copyout phases, which are strongly data/application dependent. We implemented a simple benchmark which performs a loop of 20k iterations before exiting, representative of a "medium" size GPU kernel, and is completely computation bound (no memory transfer or accesses).

We performed 100 experiments for each configuration, both for LK and traditional CUDA. We performed two sets of experiments: one where only one GPU core is used, and one where the full GPU is used. Table II shows the time (clock cycles spent on the host) to perform the aforementioned phases.

Single SM			
LK Init	LK Trigger	LK Wait	LK Dispose
509M	239	190k	30M
CUDA Alloc	CUDA Spawn	CUDA Wait	CUDA Dispose
496M	3.9k	175k	274k
Full GPU			
LK Init	LK Trigger	LK Wait	LK Dispose
503M	210	190k	30M
CUDA Alloc	CUDA Spawn	CUDA Wait	CUDA Dispose
497M	3.8k	176k	247k

TABLE II
AVERAGE VALUES FOR LK AND TRADITIONAL CUDA (SINGLE SM)

Unfortunately, while performing the former experiment, that is, with a single-SM, in some cases the GPU device got stuck. This is due to the fact that triggering a single SM means transfering only a few bytes of data across the PCI connector (the associated mailbox items), and in some cases the optimization mechanism at the driver-level indefinitely postpones this (excessively small) transfer. We therefore were forced to transfer the full mailbox also in this cases, and this explains how the numbers in the two tables are comparable. Nicely, we see that LK outperforms "standard" CUDA by a factor of 10× for the Trigger phase. This means that we are more "reactive", and potentially capable of handling finer-grained kernels (on the order of few thousands of clock cycles), because the overhead to offload work on the GPU is smaller. This is a promising result also for non-real-time GPU computing *tout court*. Unfortunately, on the other hand, the Wait phases behaves similarly (around 170k vs. 190k cycles) for LK and CUDA, as explained, because of low-level data transfer policies by the driver and OS. Further optimizations are needed at this point, and we plan to do them as a next step of the project. Init (resp. Alloc) and Dispose phase are less interesting from a point of view of LK, because they are only performed at system boot and shutdown. We anyhow notice how the latter phase is one order of magnitude slower in LK.

LK Init	LK Trigger	LK Wait	LK Dispose
521M	1.1k	203k	33M
CUDA Alloc	CUDA Spawn	CUDA Wait	CUDA Dispose
501M	7.7k	176k	893k

TABLE III
WORST VALUES FOR LK AND TRADITIONAL CUDA (SINGLE SM))

Table III shows the worst case times for the considered phases, only for the case of single SM (numbers involving the full GPU are similar). We see that both for LK and traditional CUDA, the variance is significant for the Trigger phase, while LK also suffers some variance against average performance for the Wait phase. This variance in platform performance is crucial in the real-time domain, where worst case performance (and its difference with average-case performance) must be minimized. For this reason, we will explore on low-level software optimization as a next step of the project.

IV. Conclusion and Acknowledgement

GPUs are extremely powerful machines, but they are not yet ready for adoption within industrial real-time settings. This is mainly due to their complex architecture, and non-openness of software runtime and drivers. As opposed to state-of-the-art of real-time GPU computing, we exploit intra-GPU parallelism, and provide a framework to explore real-time capabilities of most advanced architectures within the single GPU devices. The framework is not yet completed, but current results are promising, and we already identified research and development paths for our *LightKernel* tool.

This work was supported by the HERCULES Project, funded by European Union's Horizon 2020 research and innovation program under grant agreement No. 688860

References

[1] NVIDIA, "The Tegra X1 Platform," 2015. [Online]. Available: http://www.nvidia.com/object/tegra-x1-processor.html
[2] G. A. Elliott, B. C. Ward, and J. H. Anderson, "GPUSync: A Framework for Real-Time GPU Management," in *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, Dec 2013, pp. 33–44.
[3] K. Gupta, J. A. Stuart, and J. D. Owens, "A study of persistent threads style GPU programming for GPGPU workloads," in *Innovative Parallel Computing (InPar), 2012*. IEEE, 2012, pp. 1–14.
[4] N. Capodieci and P. Burgio, "Efficient implementation of genetic algorithms on GP-GPU with scheduled persistent CUDA threads," in *PAAP 2015, Nanjing, China, 2015*, 2015, pp. 6–12.
[5] "CUDA Toolkit Documentation v7.0," http://docs.nvidia.com/cuda/index.html, accessed: July, 30th 2015.
[6] Kronos Group, "The OpenCL 1.1 Specifications," 2010. [Online]. Available: http://www.khronos.org/registry/cl/specs/opencl-1.1.pdf
[7] A. Marongiu, P. Burgio, and L. Benini, "Supporting OpenMP on a multi-cluster embedded MPSoC," *Microprocessors and Microsystems - Embedded Hardware Design*, vol. 35, no. 8, pp. 668–682, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.micpro.2011.08.010

MrsP on Semi-Partitioned Systems

Jorge Garrido, Juan A. de la Puente, Juan Zamorano
Sistemas de Tiempo Real e Ingeniería de Servicios Telemáticos (STRAST)
Universidad Politécnica de Madrid (UPM)
Email: str@dit.upm.es

***Abstract*—Scheduling analysis schemes for multiprocessor platforms is a very active research area. Among other proposals, the MrsP protocol has acquired academic relevance due to its specific benefits. In general words, MrsP proposes a spin-based serialization algorithm for shared resources accesses control, emulating the priority ceiling protocol and acquiring its main properties. In this paper we preliminarily explore how the MrsP protocol could be used to build Semi-Partitioned systems.**

I. Introduction

Several different approaches have been developed for supporting multiprocessors real-time systems. However, none of them has acquired enough relevance, development and overcame the common drawbacks of these approaches. Among others, the main drawbacks of proposed multiprocessor approaches include excessive restrictions on the task model, complexity or run-time overheads.

Another point is that none of the most extensively developed approaches has aimed to be general purpose. On the contrary, all approaches have focused only on partitioned or global scheduling approaches and only a few of them on semi-partitioned systems. The same way, main approaches have been constrained to specific tasks models.

The MrsP protocol [1] was conceived to be a protocol solving the main multiprocessor approaches drawbacks without necessarily restricting the scheduling systems to be partitioned, semi-partitioned or globally scheduled, neither restricting the task model. However, only fully partitioned systems with the sporadic task model using fixed priorities are addressed in [1].

In this paper we explore a possible approach to semi-partitioned systems [2], where tasks can be statically allocated to processors or be shared among all or a set of them, using the MrsP protocol for shared resources control.

The rest of the paper is structured as follows. Section II gives a summary of the main characteristics of MrsP referenced on this paper. Then section III presents the main lines of the proposals under development, while section IV presents the preliminary scheduling proposals. Finally some conclusions and future work lines are given.

II. MrsP protocol

The MrsP protocol [1] is addressed to provide an architecturally neutral shared resources control protocol for multiprocessor systems. The protocol main contribution is the local serialization of shared resources accesses by means of a spin-busy waiting at the local priority of the resource.

This work has been partially funded by the Spanish National R&D&I plan (project M2C2, TIN2014-56158-C4-3-P).

For fully partitioned systems with tasks scheduled using fixed priorities, considering the general task model, a response time analysis is given in [1]. This response time analysis is as follows: let c^j be the maximum execution time for a resource, and $|map(G(r^j))|$ a function returning the number of processors from which the resource r^j is accessed, then the cost of accessing a resource (noted as e^j) is bounded to:

$$e^j = |map(G(r^j))|c^j \tag{1}$$

The worst case executing time of a task τ_i is the sum of its worst case execution time added the cost of each access (n_i) to a shared resource accessed by the task (returned by function $\mathbf{F}(\tau_i)$):

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j \tag{2}$$

The final response time of a task τ_i under MrsP for fully partitioned systems using the sporadic task model is:

$$R_i = C_i + max\{\hat{e}, \hat{b}\} + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{3}$$

where $\hat{e}$ stands for the maximum execution time of a resource used by a local task with a priority lower than τ_i and a task of equal or higher priority and $\hat{b}$ stands for the maximum non-preemptive execution time induced by the RTOS. Finally, $\mathbf{hpl}(i)$ returns the set of local tasks with higher priority than τ_i.

To accomplish this response time analysis, it is required that, at any given time, if a task accessing or spinning to access the resource is active, progress is done on the shared resource. This means that, if the task inside the resource is locally preempted, and another tasks is spinning at another processor, then the execution should continue on this second processor. This is proposed in [1] to be implemented by thread migration or parallel execution. None of the proposals of section III is affected by choosing any of them.

III. Proposed approaches

In this section the different approaches are motivated and presented. The general aim of these approaches are to add shared tasks to the system scheduling without fundamentally

affecting the response-time analysis of statically allocated tasks. To accomplish this, all proposals included on this section share one fundamental condition: all shared tasks are given lower priority than any statically allocated task.

Different proposed approaches are divided in two groups, those not affecting the response-time analysis of statically allocated tasks, and those having a low, bounded effect on this response time analysis.

A. Non-intrusive approach

As shown before, one of the main advantages of MrsP is the upper bounding of the number of concurrent accesses to a shared resource. This upper bound is the number of processors from where the resource can be accessed.

As a result, this safe upper bound is not affected if tasks not sharing any resource are added to the system. Furthermore, if these tasks are given lower priorities than tasks sharing resources, the response time analysis of the later ones is not affected, even if those tasks are allocated to their same processor.

Lemma 1: Response time analysis of tasks under MrsP is not affected by tasks with lower priorities not sharing resources.

Proposal 1: Tasks not sharing resources can be dynamically allocated to processors as shared tasks within Semi-Partitioned systems scheduled with MrsP without affecting the schedulability of statically allocated tasks as long as the later ones are assigned a higher priority.

With our proposal, the remaining execution capacity of each processor under MrsP may be used by shared tasks. How these tasks are scheduled is an open issue preliminarily approached in section IV.

Lemma 2: Response time analysis of tasks under MrsP is not affected if tasks not sharing resources are allowed to execute while MrsP tasks spin to access a shared resource.

This lemma assumes that the cost of releasing again the spinning task preempting the shared task is comparable to just granting access to the task while spinning. This assumption is however highly dependant on the computer architecture and the protocol implementation, requiring a more in deep investigation.

Proposal 2: Tasks not sharing resources can safely use the MrsP tasks spinning time to increase the overall remaining execution capacity.

It can be argued that *Proposal 2* fundamentally changes the underlying MrsP protocol from spinning to blocking. However, the spinning nature of the MrsP protocol is motivated by the need of avoiding priority inversions. Here we propose that only the shared tasks use the spinning time and only while the spinning task would be otherwise actively spinning (i.e. not preempted by a higher priority task). This does not change any MrsP property or fundamental underlying philosophy. This can be easily implemented by only allowing the shared tasks scheduler (whichever it is) to schedule tasks while there is no runnable task of higher priority than the local ceiling priority of the accessed resource and the status of the accessing task is spinning (i.e. it has not being granted access to the resource and the processor has not been required to undertake the work of the task inside the resource if it is locally preempted).

While this proposal addresses one of the main drawbacks of MrsP (the spinning impact on effective processor utilization), a safe bound to the executing capacity acquired by using this spin-waiting time is under further investigation. While no such safe bound seems to be reachable, tasks with firm and soft real-time requirements are still candidates to take advantage of this spinning time.

B. Sharing resources approach

While in section III-A we restricted the eligible tasks for being shared among processors to be those not using shared resources, in this section we relax this restriction. Even not affecting the higher priority (statically allocated) tasks response times is a desirable property, a much higher flexibility can be acquired with a low overhead on these tasks.

In our next proposal we keep the aim of not changing the upper bound of parallel accesses to shared resources, the $|map(G(r^j))|$ factor in the MrsP response time analysis.

Lemma 3: The $|map(G(r^j))|$ factor for each resource is not affected by the addition of tasks to a system as long as the added tasks do not use different shared resources than the already allocated tasks to processors where added tasks can execute.

Lemma 4: The addition of tasks not incrementing the $|map(G(r^j))|$ for any resource only affects the response time analysis of tasks executing on the same processor as added tasks.

Lemma 5: The addition of lowest priority tasks to a processor only affect the response time analysis of the lowest priority tasks already allocated using the same shared resources as the added task, assuming homogeneous resources access times as in [1].

Proposal 3: Shared tasks can be dynamically allocated to processors where all the resources used by the shared tasks are already used by statically allocated tasks. Shared tasks must be assigned lower priorities than any statically allocated task. Lowest priority tasks statically allocated using the resources also used by shared tasks must add the blocking time due to the shared tasks accesses to those resources.

With this proposal, only the lowest priority tasks among the statically allocated tasks are affected by the shared tasks. However, a greater flexibility is obtained, being possible to share tasks using shared resources. Again, a complete schedulability analysis is under research, with a preliminary approach proposed in section IV.

IV. Shared tasks scheduling

In sections III-A and III-B three proposals are offered on how shared tasks can be safely included in Semi-Partitioned systems based on the MrsP protocol, without compromising the schedulability analysis of the statically allocated tasks. However, the scheduling of the shared tasks is an open issue, preliminarily addressed in this section.

A. Desired characteristics

A set of desired characteristics for the scheduling of shared tasks have been identified:

- Maximize the processor utilization: the main objective of Semi-Partitioned systems is to maximize the utilization of available processor time not used by statically allocated tasks. As such, the scheduling algorithm should be aligned to this objective.
- Have an effective schedulability test.
- As far as possible, exploit the well-known and proven scheduling algorithms and properties developed for single-processor real-time systems. One of the main contributions of MrsP is its effective translation of well understood properties of single-processor resource control policies to multiprocessor systems.
- Produce as few migrations as possible. Although the main benefit of Semi-Partitioned systems arises from the possibility of executing specific tasks on more than one processor, migrations inherently pose an overhead. Thus, a scheduling algorithm in which shared tasks are more likely to complete each activation duty on a single processor is preferred.

B. Non-sharing resources tasks scheduling

An initial approach for *Proposal 1* under research is to use a Global EDF scheduler [3] to schedule the shared tasks. This approach includes some of the desired characteristics:

- Has an effective schedulability test for the task set restrictions previously explained. The maximum processor utilization available for shared tasks can be obtained by subtracting the processor utilization of allocated tasks by the response time analysis proposed in [1] to the total processor utilization available. Another possible way under investigation to calculate the execution capacity of shared tasks under the proposed systems is proposed in [4]. Finally, in general terms, the resulting system with regard to shared tasks has similarities with levels *C and D* of the system model presented in [5]. As such, the analysis given in [6] by using the service functions for each processor proposed in [7] is also being considered.
- Properties of single processor EDF are well known, as well as Global EDF has been widely studied and different variations have been proposed over time.

C. Sharing resources tasks scheduling

A scheduling scheme for *Proposal 3* is, however, far from trivial. An initial approach is again to use Global EDF for systems in which, as in the original MrsP response time analysis formulation [1], the shared resources access times are homogeneous for all processors.

Unfortunately, to the best of our understanding, there is no mature scheduling scheme that can take advantage of heterogeneous shared resources access times without adding restrictions to the task set. Heterogeneous shared resources access times could be considered for scheduling schemes ensuring executions on specific processors for specific sections of code.

V. Conclusions

MrsP protocol is a spin-based shared resource algorithm for partitioned systems in which the shared resource access is serialized by spinning at the local ceiling priority of the resource. This algorithm leads to a response time analysis similar to the PCP response time analysis for single-processors.

Here we have preliminarily analysed how MrsP could be used for Semi-Partitioned systems as long as shared tasks are given the lowest priorities. Two main approaches have been proposed, one non-invasive for allocated tasks and a second one only adding blocking time to the allocated lowest priorities tasks using each resource on each processor.

For the first approach, shared tasks are restricted to not use any shared resources and preliminarily a Global EDF scheduling is proposed. For the second approach, shared resources usage by shared tasks is allowed as long as the shared resources used are also used by statically allocated tasks on same processors. This approach slightly affects the response time analysis results for statically allocated tasks. For shared tasks scheduling, also Global EDF is preliminarily proposed.

On-going research includes formalising the presented proposals as well as further investigating the scheduling possibilities for shared tasks.

References

[1] A. Burns and A. J. Wellings, "A schedulability compatible multiprocessor resource sharing protocol–mrsp," in *Real-Time Systems (ECRTS), 2013 25th Euromicro Conference on*. IEEE, 2013, pp. 282–291.

[2] J. H. Anderson, V. Bud, and U. C. Devi, "An edf-based scheduling algorithm for multiprocessor soft real-time systems," in *Real-Time Systems, 2005.(ECRTS 2005). Proceedings. 17th Euromicro Conference on*. IEEE, 2005, pp. 199–208.

[3] S. K. Dhall and C. L. Liu, "On a real-time scheduling problem," *Operations research*, vol. 26, no. 1, pp. 127–140, 1978.

[4] S. Kato and N. Yamasaki, "Semi-partitioned fixed-priority scheduling on multiprocessors," in *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*. IEEE, 2009, pp. 23–32.

[5] M. Mollison, J. Erickson, J. Anderson, S. Baruah, and J. Scoredos, "Mixed-criticality real-time scheduling for multicore systems," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, July 2010, pp. 1864 –1871.

[6] H. Leontyev and J. H. Anderson, "Generalized tardiness bounds for global multiprocessor scheduling," *Real-Time Systems*, vol. 44, no. 1-3, pp. 26–71, 2010.

[7] S. Chakraborty, S. Künzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs." in *DATE*, vol. 3. Citeseer, 2003, p. 10190.

Towards Real-time Wireless Cyber-physical Systems

Romain Jacob* Marco Zimmerling† Pengcheng Huang* Jan Beutel* Lothar Thiele*
*Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
†Networked Embedded Systems Group, TU Dresden, Germany
firstname.lastname@tik.ee.ethz.ch marco.zimmerling@tu-dresden.de

***Abstract*—One big challenge to be overcome before the successful deployment of wireless cyber-physical systems is to provide *hard real-time guarantees*, not only within the wireless network, but in fact between *end-to-end application processes*. To achieve this, we design a distributed real-time protocol (DRP) that considers the complete transmission chain, including application tasks, peripheral busses, memory accesses, networking interfaces, and wireless real-time protocol. DRP guarantees that end-to-end message deadlines are met, while being adaptive to unpredictable system changes, by establishing at run-time a set of contracts among the different elements of the transmission chain.**

I. Introduction

Over the past decade, a tremendous amount of work has been carried out around low-power wireless communication technologies. Especially, wireless sensor networks (WSNs) have received much attention. One major challenge yet to be overcome is to enable reliable and efficient use of low-power wireless networking Cyber-Physical Systems (CPS), sometimes referred to as wireless sensor and actuator networks. As many CPS applications are mission-critical and physical processes evolve as a function of time, the communication among the sensing, actuating, and computing elements in CPS is often subject to real-time requirements (*e.g.*, to guarantee the stability of feedback loops). Such real-time requirements are key to enable *safe* CPS, which is arguably one of the most important features for a successful deployment of CPS [1].

Challenges. These real-time requirements are often specified from an *end-to-end application* perspective. For example, a control engineer may require that sensor readings taken at time t_s are available for computing the control law at $t_s + \mathbf{D}$. Here, the *relative end-to-end deadline* $\mathbf{D}$ is derived from the activation times of *application-level tasks*, namely the sensing and control tasks, which are typically executed on physically distributed devices. Meeting such hard end-to-end deadlines is non-trivial, because data transfer between application-level tasks involves multiple other tasks (*e.g.*, operating system, networking protocols) and shared resources (*e.g.*, memories, system busses, wireless medium). Therefore, the entire transmission chain, involving application tasks, peripheral busses, memory accesses, and wireless networking protocol, must be taken into account to tackle this challenge. We argue that doing so requires combining three building blocks:

1) on the node level, a decoupling of application (*e.g.*, sensing, actuation, control) and wireless communication tasks using a *dual-processor architecture*;
2) on the network level, an efficient *wireless real-time protocol*, which guarantees that messages between source and destination nodes are delivered reliably in real-time;
3) overall, a *distributed real-time protocol* that manages the flows of messages across the network, decouples responsibilities between components, and ensures that end-to-end deadlines *between application tasks* are met.

In this paper, we first briefly present the design of our solution (Sec. II), then we highlight remaining problems and how we intend to address them in our ongoing work (Sec. III).

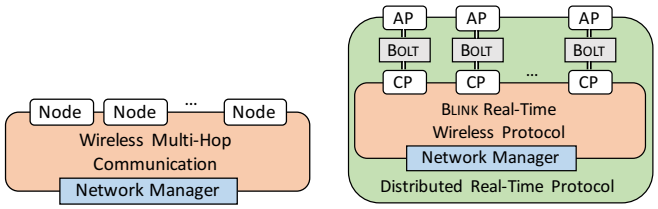
II. Overview of the solution

We consider that all communications between a source and destination node are subject to real-time constraints. We call one such communication a *flow*. Let $\mathcal{F}$ be the set of all real-time message *flows* in the system. Each flow $F_i = (s_i, d_i, T_i, \mathbf{D}_i)$ is defined by a *source node* s_i, that *releases* messages with a *minimum inter-message interval* T_i, also called *period*. Every message released by s_i should be delivered to the *destination node* d_i within the same *relative end-to-end deadline* $\mathbf{D}_i$. No message can be sent outside of a flow, and each flow must be *registered* (*i.e.*, accepted by the protocol) before it can start releasing messages. Several flows, eventually with different period and deadline, may be registered between the same source and destination nodes.

The wireless network consists of a set of *nodes* $\mathcal{N}$ that exchange messages via *wireless multi-hop communication*, as illustrated in Fig. 1(a). A logically global *network manager* arbitrates access to the shared networking resource. Physically, the network manager may run on one of the nodes. The overall design is based on three building blocks, as described next.

Dual-processor architecture. When using the traditional system architecture shown in Fig. 1(a), application and communication tasks execute concurrently on each node. When both tasks attempt to simultaneously access shared resources (*e.g.*, memory, processor, system bus), one of them will be delayed for an arbitrary time. Such interference hampers timing predictability, making real-time guarantees difficult to provide.

To tackle this issue, we propose to replace each node with a dual-processor platform. One processor (*AP*) runs only the application, while the other processor (*CP*) executes only the wireless multi-hop communication protocol. Using the Bolt interconnect [2], AP and CP are decoupled in time, power, and clock domains, and can asynchronously exchange messages with bounded delay, with ultra-low-power overhead.



(a) A set of nodes, each with a single processor, execute the application and exchange messages via wireless multi-hop communication.

(b) Application (AP) and communication (CP) processors exchange messages via Bolt, while the CPs run the Blink real-time wireless protocol.

Figure 1. Traditional (a) and our proposed (b) system architecture. *A logically global network manager arbitrates access to the shared wireless medium.*

Wireless real-time protocol. Providing real-time guarantees across wireless networks is challenging. In particular, to support real-world CPS applications, one needs a protocol that delivers packets reliably (*i.e.*, with high reception rate) within real-time deadlines, while being adaptive to dynamic changes in the wireless network and traffic demands.

Out of the many solutions that have been proposed over the years, Blink [3] is the only wireless real-time protocol that satisfies such requirements. It is built on top of LWB [4] and leverages fast and highly reliable Glossy floods [5], a protocol based on synchronous transmissions, which represents a paradigm shift away from traditional routing-based protocols.

Distributed real-time protocol. The use of Bolt to decouple the application (running on the APs) and the Blink wireless real-time protocol (running on the CPs) brings necessary benefits, such as flexibility in the operation mode for each component (*e.g.*, time- versus event-triggered) and interference mitigation. But at the same time, it also triggers a significant challenge: while the APs and CPs should execute independently to prevent interference, it is their *joint* operation that determines whether messages exchanged between the APs meet their end-to-end deadlines.

To address this challenge, we introduce a distributed real-time protocol (DRP), which represents the third building block of our solution. DRP strikes a balance between the decoupling of AP and CP/Blink on the one hand and the end-to-end timing predictability of message exchanges between the source and destination APs of one flow on the other hand. This trade-off is embodied by mutual *contracts*. A contract settles the least required agreement between APs, CPs and Blink such that all can operate as much as possible independently, while ensuring that end-to-end message deadlines are met. DRP establishes contracts at runtime, as flows are dynamically requested and removed, and scales efficiently to large sets of real-time message flows.

Using these three building blocks, the overall system architecture evolves from Fig. 1(a) to Fig. 1(b). In the remainder of this section, we detail the design of DRP and how contracts are established when a new flow is requested at runtime.

DRP: Distributed Real-time Protocol

DRP uses contracts that are dynamically established at runtime to provide end-to-end real-time communication between the source and destination node's AP of every flow. This includes guaranteeing that message buffers along the whole transmission chain never overflow.

In a nutshell, the overall end-to-end latency of a message depends on how often APs and CPs read out messages from Bolt, and the maximal delay for a message to be served by Blink. This can be formalized by three parameters for each flow F_i: the *flushing periods of the source* and *destination* nodes, and the *network deadline* of F_i, denoted by T_f^s, T_f^d, and D_i respectively. The network deadline D_i is computed online by the source node's AP when a new flow is requested. It represents the deadline which is requested to Blink (*i.e.*, if Blink accepts this new flow, it guarantees that any message is delivered at the destination node's CP within D_i).

DRP decides on the *distribution of responsibilities* among the source node, Blink, and the destination node of a flow F_i with regard to meeting its end-to-end deadline $\mathbf{D}_i$ using a configuration parameter of DRP, the *deadline ratio* r, chosen at design time. The source node and Blink are jointly responsible for meeting a fraction r of the end-to-end deadline $\mathbf{D}_i$,

$$f(T_f^s, D_i) = r * \mathbf{D}_i \tag{1}$$

The remaining part of the overall end-to-end deadline determines the responsibility of the destination node,

$$g(T_f^d) = (1 - r) * \mathbf{D}_i \tag{2}$$

One can derive concrete expressions for $f(\cdot,\cdot)$ and $g(\cdot)$ after a detailed worst-case timing and buffer analysis of the system.

Overall, DRP dynamically establishes two contracts for each newly admitted flow $F_i \in \mathcal{F}$ in the system:

Source $\leftrightarrow$ Blink F_i's source node s_i agrees to write no more messages than specified by the flow period T_i, and prevents overflows of Bolt and CP_s's local message buffer. In turn, Blink agrees to serve F_i such that any received message meets the network deadline D_i.

Blink $\leftrightarrow$ Destination Blink agrees to deliver no more messages than specified by T_i. In turn, F_i's destination node d_i agrees to read out all delivered messages such that overflows of Bolt and CP_d's local buffer are prevented and all messages meet the flow's end-to-end deadline $\mathbf{D}_i$.

For any flow, if both contracts are fulfilled, all messages that are successfully delivered by Blink will meet their end-to-end deadlines. In practice, the fulfillment of contracts is guaranteed by a set of *admission tests*, which are performed in sequence upon registration of a new flow. The overall mechanism of contracts (*i.e.*, sharing of responsibility, flow registration, and admission tests) is illustrated on Fig. 2.

In our ongoing work, we have derived the theoretical optimal performances that can be provided by DRP, in terms of responsiveness (*i.e.*, smallest end-to-end deadline) and bandwidth that can be supported. We also evaluated the run-time behavior of DRP in simulation, based on values and parameters from physical implementations of both Bolt and Blink. Our results show that the end-to-end latency of messages can be up to 96% of the our analytic worst-case

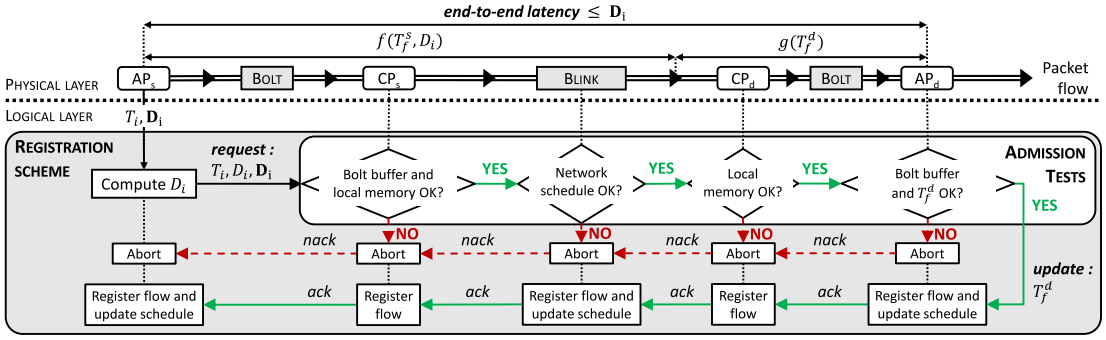


Figure 2. Steps and components involved when registering a new flow in DRP. *Given a request for a new flow $F_i = (s_i, d_i, T_i, \mathbf{D}_i)$, the source node's AP computes the flow's network deadline D_i. Then, all components successively verify using specific admission tests whether they can admit the new flow. DRP registers a flow only if all admission tests succeed, which triggers changes in the runtime operation (i.e., schedule) of some components.*

bounds. Thus, since our model of DRP's performance is safe and tight, it can be leveraged for system design.

III. Open Questions and Future Work

As we mention in the introduction, DRP is an initial step, and requires further investigations before we can claim it efficiently solves the real-time challenge in wireless CPS. In particular, we detail thereafter three key open questions that we intend to investigate in future work.

Physical implementation of DRP. Even though our preliminary simulation results are encouraging, they are not sufficient to validate the suitability of the protocol in a real-life setting. As illustrated in Fig. 1(b), DRP requires specific hardware (*i.e.*, a Bolt-enabled wireless network). Our group recently designed and produced custom-built dual-processor boards where Bolt interconnects a powerful application processor (TI MSP432) with a state-of-the-art communication processor (TI CC430). These have been implemented on the Flocklab testbed [6] and this new network will be publicly available for testing soon.

Leveraging this experimental setting, we plan to implement DRP and extensively test it to validate the registration and de-registration of flows at run-time, experiment with the failure and recovery of nodes, and verify that end-to-end real-time guarantees hold in such dynamic scenarios. The accurate time synchronization of FlockLab will allow us to validate the analytic worst-case delay analysis in a real system.

Dependability. Blink is built on the state-of-art wireless protocol LWB which achieves more than 99.9% data yield [4]. However, packet loss may still happen, and the sensitivity of DRP to these losses must be investigated. While a moderate loss of data packets can often be tolerated by the application, loss of schedule packets, that drive the operation of Blink, may be more critical.

We need now to develop a retransmission and/or dependability scheme for DRP to mitigate such effects and provide (at least) probabilistic guarantees for a safe behavior of the overall protocol, using *e.g.*, probabilistic model-checking. Some inspiration may be found in [7].

Reaching meaningful performances. Finally, the goal of this work is to enable practical implementations of wireless CPS. To this end, DRP must meet relevant latency requirements (*e.g.*, as mentioned in [1]). This means flow periods and end-to-end deadlines ranging from tens of milliseconds to half a second.

The question to investigate is how a given implementation of DRP can be optimized to meet such requirements. Is this even possible? If not, where do the main limitations come from? How can we optimize our design to overcome such limitations? In our opinion, the main trade-off comes from the decoupling between the various components. It brings flexibility and mitigates interference, but at the cost of a larger minimum end-to-end deadline that can be supported.

IV. Conclusions

Concealing hard real-time guarantees and wireless communication is very challenging. However, the emergence of Glossy-based protocols help significantly, as they eliminate the need for complex routing and enable unparalleled flexibility and adaptability in low-power wireless communications. Hence, we are striving to bridge this gap and eventually enable the successful deployment of safe and reliable wireless CPS.

Acknowledgments. This work was supported by Nano-Tera.ch, with Swiss Confederation financing, and by the German Research Foundation (DFG) within the Cluster of Excellence "Center for Advancing Electronics Dresden" (CFAED).

References

[1] J. Åkerberg, M. Gidlund, and M. Björkman, "Future research challenges in wireless sensor and actuator networks targeting industrial automation," in *Proc. of IEEE INDIN*, 2011.

[2] F. Sutton, M. Zimmerling, R. Da Forno, R. Lim, T. Gsell, G. Giannopoulou, F. Ferrari, J. Beutel, and L. Thiele, "Bolt: A stateful processor interconnect," in *Proc. of ACM SenSys*, 2015.

[3] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele, "Adaptive real-time communication for wireless cyber-physical systems," ETH Zurich, Tech. Rep., 2016.

[4] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *Proc. of ACM SenSys*, 2012.

[5] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *Proc. of ACM/IEEE IPSN*, 2011.

[6] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Proc. of ACM/IEEE IPSN*, 2013.

[7] M. Zimmerling, F. Ferrari, L. Mottola, and L. Thiele, "On modeling low-power wireless protocols based on synchronous packet transmissions," in *Proc. of IEEE MASCOTS*, 2013.

Preemption Point Selection in Limited Preemptive Scheduling using Probabilistic Preemption Costs

Filip Marković, Jan Carlson, Radu Dobrin
Mälardalen Real-Time Research Center, Mälardalen University, Sweden
{filip.markovic, jan.carlson, radu.dobrin}@mdh.se

***Abstract*—Limited Preemptive Scheduling is an attractive paradigm that enables controlling preemption related overheads, by appropriate preemption point selection. The selection of preemption points is essential to ensure schedulability and the associated analysis accounts for upper bounded preemption overheads, thus introducing a potentially high level of pessimism in the results. In this paper we propose a probabilistic distribution model of preemption related overhead and an accompanying method for preemption point selection based on quantiles, which provides controllable probabilistic relaxations. An experimental evaluation demonstrates the improvement of the extent to which this new approach facilitates finding solutions to the preemption point selection problem.**

I. Introduction

Limited Preemptive Scheduling (LPS) is an attractive alternative to non-preemptive scheduling or fully preemptive scheduling with respect to, e.g., addressing real-time systems schedulability, or efficiently controlling the number of preemptions and associated overheads. LPS can efficiently reduce the blocking by lower priority tasks, compared to non-preemptive scheduling, as well as generate a significantly lower number of preemptions compared to the fully preemptive scheduling.

The significance of the preemption related overheads in LPS, and real time systems in general, has received an increased attention during the past decade [1], [2]. The preemption related overheads are traditionally derived from the code level analysis taking into account, e.g., the Cache Related Preemption Delays (CRPD), pipeline and instruction cache effects, or context switch costs [3], and further used to analyse their impact on the LPS schedulability [4]. Consequently, the current state of the art relies on overly pessimistic assumptions about CRPDs. Hence there is a need for new, unified approaches that benefit from a better use of information from the code analysis in the schedulability analysis and vice versa.

Unified code level and schedulability level analysis is mainly aiming at bounding the preemption costs and reducing the level of pessimism in the computation of preemption-related overheads [5], [6]. In the context of LPS, the existing approaches typically assume that the upper bounds of preemption related delays are provided from the code-level analysis (e.g., CRPD analysis), and that the possible preemption points are predetermined by the programer, out of which a set of them is selected to guarantee schedulability [4], [7]. However, the schedulability analysis is overly pessimistic as it is based on upper bounded preemption overheads, which at runtime are typically generated with a very low probability. Hence, the analysis can benefit from a probabilistic approach that takes into account the probabilistic distribution of the preemption related overheads at runtime, and thus enlarges the number of sets of feasible preemption point selections.

Empirical methods can be used to estimate probabilistic distributions of the preemption overheads. Bastoni et al. [2] proposed two approaches for measuring CRPD – *a schedule-sensitive method*, that measures scheduler-dependent effects of cache, and *a synthetic method*, that can quickly record a large number of measured samples. Thus, both approaches provide information about the statistical distributions and average CRPDs, as well as cache-related preemption and migration delays (CMPDs). The derived probability distributions of the preemption costs enrich the analysis with additional information that provides for efficient preemption point selection and associated probabilistic schedulability guarantees.

We envision a framework based on a probabilistic preemption overhead model that enables preemption point selections with associated probabilistic schedulability analysis under the LPS paradigm. As a first step, in this paper we propose a probabilistic distribution model of preemption related overheads and, building upon the work of Bertogna et al. [8] (LiP-opt), we propose a new preemption point selection approach based on quantiles. Our approach provides preemption point selections that guarantee taskset schedulability in all cases where the LiP-opt method does, but it also finds selections for some task sets, deemed as unfeasible by LiP-opt, by gradually relaxing the upper bounds on preemption overheads according to the probabilistic distributions.

II. System Model

In this Section we describe the task model that builds on the work presented by Bertogna et al. [8], and our probabilistic preemption overhead model.

A. Task Model

We consider a set of sporadic tasks Γ, composed of n real-time tasks under uni-processor Fixed Priority (FP) paradigm. Every task τ_i $(1 \leq i \leq n)$ has a worst case execution time C_i and generates an infinite number of task instances, with the first instance arriving at any time and the successive instances are separated by minimum interarrival times T_i. The relative deadline D_i of each task is less than or equal to the task period $(D_i \leq T_i)$. Furthermore, each task consists of a sequence of m non-preemptive blocks $B_{i,k}$, $(1 \leq k \leq m)$, with worst

case execution times $C_{i,k}$, $C_i = \sum_{k=1}^{m} C_{i,k}$. Consequently, the blocks of a task τ_i are separated by $m-1$ possible preemption points denoted by $PP_{i,k}$. The preemption overhead upper bound associated with $PP_{i,k}$ is denoted by $PO_{i,k}$.

B. Probabilistic Overhead Model

In this paper we use a model of preemption related overheads based on probability density functions. This model enhances the information level of the preemption overhead compared to the existing single valued upper bound models. We assume that the overhead distribution of each possible preemption point $PP_{i,k}$ is represented by a *probability density function* $PDF_{i,k}$ with distinct upper bound $PO_{i,k}$ such that $\forall x \mid x < 0 \vee x > PO_{i,k} \Rightarrow PDF_{i,k}(x) = 0$ (see Figure 1).

The overhead distributions can be derived using various methods e.g., empirical measurements, CRPD analysis, or using probabilistic convolution of multiple methods. For example, in order to combine empirically derived results (marked red) with a pessimistic upper bound from static code analysis (marked black) they are primarily joined in a union sample. By sorting the sample for $PP_{i,k}$ on the preemption related overhead axis we can use the *density estimation methods* (e.g., Kernel density estimation) to estimate the $PDF_{i,k}$.

From $PDF_{i,k}$ we derive a quantile preemption overhead value using the quantile function. The quantile function $Q(p, PDF_{i,k})$ for a given probability p in the probability distribution $PDF_{i,k}$ calculates the preemption overhead value such that the probability of the overhead being less or equal to the calculated value is equal to the given probability p. Since we assume that the upper bound is always included in the overhead distribution, consequently $Q(1, PDF_{i,k}) = PO_{i,k}$.

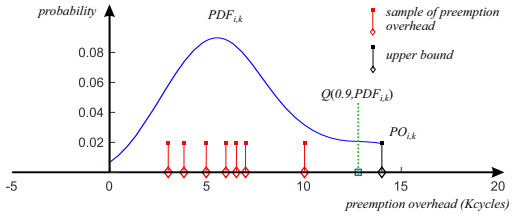


Fig. 1. Probabilistic overhead model derived from the empirical samples

III. Selection Approach

The preemption point selection approach in this paper is based on the LiP-opt method proposed by Bertogna et al. [8] and exploits the benefits of using a probabilistic overhead model. The goal is to select a subset of selected preemption points (SPP) from the set of possible preemption points and enable them for runtime preemption. After the selection, the interval between consecutive SPPs is called a non-preemptive region (NPR). The LiP-opt method computes an optimal preemption point selection with respect to the minimisation of the preemption overhead per task while guaranteeing schedulability. If a feasible selection is found, by computing the maximum NPR for each task based on execution times (C_i) of its higher priority tasks, then the schedulability of the task set is guaranteed. C_i is recalculated after each selection by including the preemption overheads of the selected points in the updated execution time ($C'_i = C_i + \sum PO_{i,k} \mid PP_{i,k} \in SPP$). Therefore, the selection algorithm is computed in decreasing priority order from τ_1 to τ_n. For example, for task τ_i the computation of NPR_i includes C'_j $(j < i)$ of all higher priority tasks. The algorithm will not find a feasible selection *if* $\exists\, i, k$ *such that* $NPR_i < PO_{i,k} + C_{i,k}$.

By using our probabilistic preemption overhead model (instead of only upper bounds) we increase the number of possibilities for the preemption point selection. Importantly, the quantile approach can overcome the NPR infeasibility condition by providing relaxation not only in the preemption overhead, but also in the run-time utilisation (by selecting points which are less likely to produce high overheads).

In order to achieve the above mentioned benefits, we propose a *quantile selection* approach. Selection based on quantiles allows us to consider the overhead associated with a preemption point at a given confidence level p. With $p = 1$, the quantile function returns the upper bound values, while setting p such that $p < 1$ provides different quantile values for the different preemption distributions, thus implying a different point selection.

$p \leftarrow 1$
while $(p \geq threshold)$ **do**
  for $i \leftarrow 2$ **to** n **do**
    $PO^p_{i,k} \leftarrow Q(p, PDF_{i,k})$ for each $PP_{i,k}$ of τ_i
    compute NPR_i using C'_j for $j < i$
    $SPP^p_i \leftarrow$ PP selection w.r.t. $PO^p_{i,k}$ and NPR_i
    $C'_i \leftarrow C_i + \sum PO^p_{i,k} \mid PP_{i,k} \in SPP^p_i$
  end
  if $(\forall i\ SPP^p_i \neq infeasible)$ **then**
    return $\bigcup_{2 \leq i \leq n} SPP^p_i$
  else
    $p \leftarrow p - \epsilon$
  end
end
return *infeasible*

Algorithm 1: The quantile selection approach

In first iteration, p is set to 1 (see Algorithm 1), and the preemption point selection will be exactly the same as with LiP-opt – considering the upper bounds and satisfying hard schedulability guarantees. However, if a feasible selection cannot be computed, p is decreased by the predefined amount ϵ and a new attempt is made to find a selection, this time based on the $p - \epsilon$ quantile overhead values, thus relaxing the NPR_i value and C'_i of tasks. The algorithm stops with the first quantile solution that satisfies relaxed schedulability guarantees or when the predefined threshold is reached.

IV. Experimental Results

In order to investigate the applicability of the proposed approach, we have conducted a number of experiments to calculate the ratio of schedulable task sets for which a feasible preemption point selection is found based on preemption overhead upper bounds, against our *quantile selection* approach where the considered overheads are based on a gradually

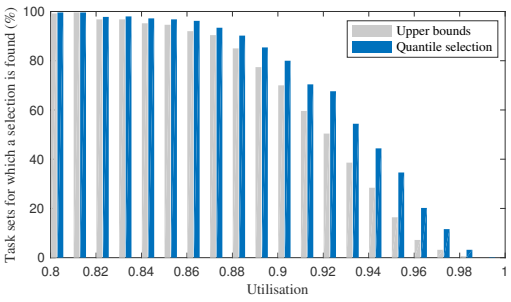


Fig. 2. Upper bound- and quantile-based preemption point selections

decreasing probability factor. The purpose of the experiments was to investigate to what extent relaxing the considered overheads facilitates finding solutions to the preemption point selection problem.

In the experiment setup we generate 500 synthetic task sets, consisting of 7 tasks per task set. Each task consists of a random number of BBs, in the range [20, 200], with the WCET of each BBs generated according to a Gaussian distribution ($\mu = 4000$, $\delta = 3000$). The upper bounds of the preemption overhead are generated according to the distribution described by Bertogna et al. [8], for each possible preemption point. Based on these upper bounds (denoted u for brevity below), the probabilistic distributions for the preemption overheads are randomly generated from three possible Gaussian distributions, selected with equal probability: ($\mu = \frac{3}{20}u$, $\delta = \frac{1}{15}u$), ($\mu = \frac{1}{2}u$, $\delta = \frac{1}{15}u$), and ($\mu = \frac{3}{20}u$, $\delta = \frac{14}{15}u$). Next, we generate variants of these 500 task sets for a number of utilisation values ranging from 0.5 to 0.99 by deriving task periods with the U-unifast algorithm [9] to achieve the desired utilisation. However, the relevant results are found in the [0.8,0.98] range.

For each task set we run the selection method [8] based on preemption overhead upper bounds. We also run the proposed *quantile selection* approach, with a threshold parameter of 0.8, meaning that p values from 1 to 0.8 are considered.

Figure 2 presents the experimental results, focusing on the percentage of task sets for which a selection is found. For utilisations lower than 0.8, both approaches find selections that guarantee schedulability based on the upper bounds (in the first iteration of the *quantile selection* approach). For utilization greater than or equal to 0.8, there are task sets for which no feasible selection exists using the upper bounds, but for which the proposed *quantile selection* approach can find selections when considering gradually relaxed overhead bounds between the 1 and 0.8 quantiles. For example, when the utilisation is 0.92, the *quantile selection* approach finds a valid selection for 18% of the task sets in addition to the 50% feasible selections found based on upper bounds.

V. Conclusion and Future Work

In this paper we proposed a novel approach for preemption point selection in LPS, based on preemption related overheads modelled as probabilistic distributions. We use a *quantile based* preemption point selection that provides a wider set of feasible solution in terms of feasible preemption points, compared to traditional upper bound based approaches, thus enabling its use in different scenarios.

For systems that are already schedulable using traditional upper bound based approaches, our method will also find an optimal set of preemption points. In addition, for many systems deemed as unschedulable due to high preemption overheads, the method can still provide a set of selected preemption points by gradually relaxing the upper bounds on preemption overheads based on the probabilistic distributions.

As the next step, we plan to investigate how the proposed *quantile based selection* impacts on the probability of missing a deadline. We envision an algorithm that will heuristically derive a set of points that minimises the probability of a deadline miss. Consequently, we will investigate *quantile based selection* and the associated probabilistic schedulability analysis, as well as improving existing schedules with respect to, e.g., reducing the overheads and thus providing for more slack, improved preemption point selection with respect to critical sections and the need of synchronisation mechanisms, etc. Another line of future work considers the improvements for specified attributes of schedulable systems, e.g., to reduce the average response time or number of preemptions. Finally, the introduced probabilistic model will be enriched with more information provided by code level analysis.

Acknowledgment

We want to express our gratitude to the EUROWEB+ project that partly founded the research and also to Abhilash Thekkilakattil for valuable discussions.

References

[1] W. Lunniss, S. Altmeyer, R. I. Davis *et al.*, "A comparison between fixed priority and EDF scheduling accounting for cache related pre-emption delays," *Leibniz Transactions on Embedded Systems*, vol. 1, no. 1, 2014.

[2] A. Bastoni, B. Brandenburg, and J. Anderson, "Cache-related preemption and migration delays: Empirical approximation and impact on schedulability," *Proceedings of OSPERT*, pp. 33–44, 2010.

[3] C.-G. Lee, J. Han, Y.-M. Seo, S. L. Min, R. Ha, S. Hong, C. Y. Park, M. Lee, and C. S. Kim, "Analysis of cache-related preemption delay in fixed-priority preemptive scheduling," *IEEE Transactions on Computers*, vol. 47, no. 6, pp. 700–713, 1998.

[4] G. C. Buttazzo, M. Bertogna, and G. Yao, "Limited preemptive scheduling for real-time systems. A survey," *industrial Informatics, IEEE Transactions on*, vol. 9, no. 1, pp. 3–15, 2013.

[5] H. Ramaprasad and F. Mueller, "Tightening the bounds on feasible preemptions," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 10, no. 2, p. 27, 2010.

[6] S. Altmeyer and C. Burguiere, "A new notion of useful cache block to improve the bounds of cache-related preemption delay," in *Real-Time Systems, 2009. ECRTS'09. 21st Euromicro Conference on*. IEEE, 2009, pp. 109–118.

[7] B. Peng, N. Fisher, and M. Bertogna, "Explicit preemption placement for real-time conditional code," in *Real-Time Systems (ECRTS), 2014 26th Euromicro Conference on*. IEEE, 2014, pp. 177–188.

[8] M. Bertogna, O. Xhani, M. Marinoni, F. Esposito, and G. Buttazzo, "Optimal selection of preemption points to minimize preemption overhead," in *Real-Time Systems (ECRTS), 2011 23rd Euromicro Conference on*. IEEE, 2011, pp. 217–227.

[9] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.

Cyber-OF: An Adaptive Cyber-Physical Objective Function for Smart Cities Applications

Med Ghazi Amor[§‖] Anis Koubaa[§*], Eduardo Tovar[§], Mohamed Khalgui [‖]
[§]CISTER/INESC TEC and ISEP-IPP, Porto, Portugal
[*] Prince Sultan University, Saudi Arabia.
[‖] INSAT, University of Carthage, Tunisia.
Email: (mhgha, aska, emt)@isep.ipp.pt, akoubaa@psu.edu.sa, khalgui.mohamed@gmail.com

***Abstract*—RPL is the standard routing protocol for the Internet of Things. It is designed for low-power and lossy networks. Several works designed different objective functions for RPL to optimize routing decisions for a particular category of applications. However, these objective functions do not take into account the cyber-physical properties of the environment. In addition, they are tailored to satisfy a particular application requirement (e.g. energy efficiency or delay), so are not adaptive to possible changes of data criticality. This paper improves on the state-of-the-art with the design of a cyber-physical objective function tailored for smart city applications, that addresses the aforementioned gaps. Initial simulation results demonstrate the effectiveness of Cyber-OF in coping with dynamic changes of the criticality of events data and in providing a good performance trade-off between conflicting performance metrics, namely energy**

I. Introduction

The Internet-of-Things (IoT) is increasingly gaining popularity in both academia and industry enabling a large number of applications that integrate both the cyber-world and the digital world, namely the Internet. Gartner predicts that the value add of the IoT by 2020 would reach $1.9 Trillion whereas CISCO estimates to reach 50 billions devices by 2020 [1]. Smart cities is one of the most promising applications of the IoT and according to IoT Analytics, it is considered as the second most popular application in 2015 [2]. One of the most influencing driving factor of the IoT is the development of standard protocols stack that copes with IoT applications requirements in terms of scalability, energy-efficiency, Quality of Service (QoS) and security, including the IEEE 802.15.4 protocol and its variants for lower communication layers, and then its integration to the Internet through the 6LoWPAN initially, and later with RPL routing protocol [3] at the network layer. CoAP and MQTT were proposed as alternatives to HTTP for Internet application layer and transport layer protocols in the IoT.

RPL is a source-based distance-vector routing protocol that was designed for low-power and lossy networks, such as wireless sensor networks. RPL attracted a lot of attention in the literature considering the open design of its objective function that is responsible for shaping the routing decision. Several works (e.g. [4] [5] [6] [7] [8]) have been proposed around the specification of objective functions that improve over those specified in the standard namely OF0 and MRHOF based on ETX metric. However, most of these objective functions adopt a static behavior that intends to optimize either a single-metric or multi-metric objective function without being adaptive to the events data carried out. In addition, the proposed objective functions in the literature do not consider the cyber-physical properties of the environment such as climate conditions, that in the context of a smart city applications, may infer about the criticality of an event (e.g. high temperature would mean a fire event).

To illustrate the problem, consider an RPL-based sensor network for weather and climate conditions monitoring. Typical objective functions would be designed to optimize a certain metric of interest such as energy consumption, or delay, or throughput, or hop count, etc. Some others like in [6] proposed a fuzzy logic objective function that combines several metrics of interest like energy consumption, or delay using fuzzy rules. However, these objective functions would behave exactly the same as if they carry a normal data packet or a critical-event data packet, which might not be appropriate. In fact, in case of normal conditions, it is wiser to focus more on optimizing the energy consumption, however, when a critical event occurs, (e.g. a fire) it would be more appropriate to optimize the end-to-end delay. Thus, an adaptive behavior would fit better this dynamic nature of events in the context of IoT applications in general and smart cities applications in particular. This represents the main motivation of this paper, where we contribute with the design and development of a new objective function that (1) takes into account the cyber-physical properties of the environment, (2) provides an adaptive behavior based on the criticality of the event. We also provide initial simulation results that demonstrate the effectiveness of our approach.

The remainder of the paper is organized as follows. Section II presents an overview of previous works on design of objective functions of RPL and contrast it against the proposed cyber-physical objective function. Section III presents the cyber-physical objective function. Simulation study and performance evaluation are presented in Section IV. Section V concludes the paper and outlines future works.

II. Related Work

Routing metrics and objective functions are the responsible features for the Directed Acyclic Graph (DAG) construction in RPL. However, the standard defined by the IETF in [4] did not impose any routing metric to use. Thus, the parent

selection is implementation-specific which makes it an open research issue worth of being investigated. In this section, we briefly review the Objective Functions (OF) proposed in the literature.

In [5], MRHOF was proposed. It is an objective function based on the ETX metric which is the number of transmissions a node expects to make to a destination in order to successfully deliver a packet. It uses a metric container to specify the routing objects which are located in a DIO packet. Besides the ETX, MRHOF may be used with any routing metric defined in RFC 6551.

In [10], the authors defined a new extention for RPL called Co-RPL. It is designed for mobile low power and lossy WSN. Its main purpose is to maintain the connectivity between the nodes while providing QoS guarantees in a mobile network. Their solution is to modify the trickle timer which will depends on the speed and mobility of the node.

In [11], the authors tackled the problem of using one metric to construct the DAG and to optimize paths to the root. They considered four routing metrics for their solution to select the best neighbor. It consists on combining the Hop Count, End-to-End delay, Energy and the ETX (expected transmission count) using an artificial intelligence technique which is the fuzzy logic. This algorithm will convert these links and node metrics into one output value which will decide whether the neighbor parent deserves to be a preferred parent or no.

In summary, when studying the mentioned OFs, we notice that they do not take in consideration the cyber-physical properties of the environment. Therefore, relying on one metric or more in a critical condition (storm, fire, disaster ...) may be inefficient and does not satisfy the requirements of the smart cities application profiles. For example, the use of the hop-count metric in an emergency situation may not choose the fastest way to advertise the network. In addition, the use of one static objective function would not fit the requirements of the same applications having different types of event criticality.

III. Cyber-Physical Objective Function

We designed the Cyber-Physical objective function (Cyber-OF) to adapt the network tree structure in real-time to the cyber-physical properties of the environment based on the event criticality. In fact, for normal data packets, the objective is to maximize the network lifetime, thus, the objective function optimizes the energy consumption. In case of a critical event, the network should adapt its topology to minimize the end-to-end delays. Therefore, we adopt an adaptive behavior by considering two routing objective functions, each uses a particular metric of interest, namely:

- **Energy Metric:** this metric represents the energy consumption in an RPL node. With this metric, it is possible to extend the network lifetime. It is essential to consider this metric for applications with energy-efficiency concerns.
- **End-To-End delay:** The end-to-end delay is the average time taken by a packet to be sent from node to sink. This metric should be minimized for applications that require real-time guarantees.

The flowchart in the Figure 1 summarizes the operations of the cyber-physical objective function.

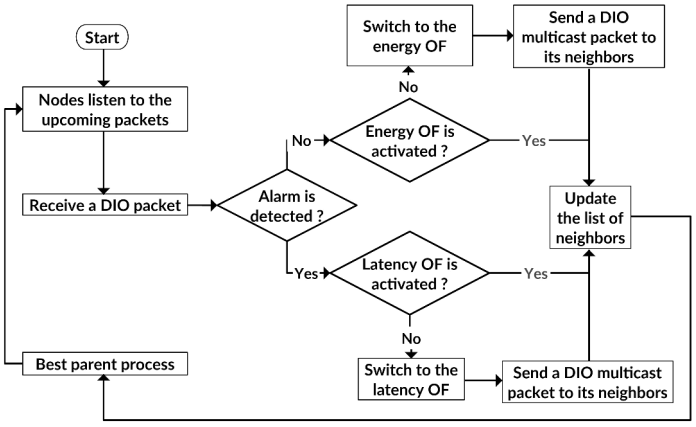


Fig. 1. Flowchart of the Cyber-Physical objective function

First, the objective function (OF) based on the energy metric is activated. Second, RPL switches to the latency OF only if a critical event is detected which will allow RPL to find the stable minimum-latency paths. The metric used by the Cyber-OF is determined by the metrics located in the DIO metric container.

For example, in normal conditions, the objective function maximizes the network lifetime. In case of an emergency situation where a critical event is detected, nodes involved in forwarding this event to the border router must use a new objective function that reduces the end-to-end delays.

IV. Simulations and Performance Evaluations

A. Environmental Setup

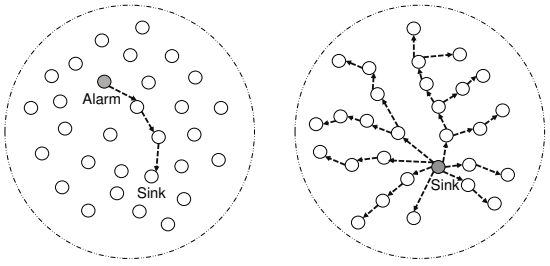


Fig. 2. Simulation scenario : (i) Send the alarm to the sink node (ii) Send a multicast packet to the whole network

We implemented the Cyber-OF in ContikiOS and we used Cooja simulator to evaluate its performance. We evaluated the performance of three implemented objective functions, namely: (1) latency-based objective function (OF), which only optimizes the latency, (2) energy-based OF, which only optimizes the energy, and (3) the Cyber-OF, which implements the adaptive behavior defined in Figure 1.

The simulations scenarios were performed using a 2D-grid surface of a network topology with 10, 20 and 30 sensors. The DAG architecture is composed of one Border Router, which represents the data sink, and the rest are UDP servers generating the data. The depth of the formed DAG is equal to 6. In this simulation, we assumed that a fire alarm will be triggered in node 10, which will send a unicast DIO packet that contains the alarm to the sink. After that, the sink will send the received alarm to all nodes of the DAG to adapt the topology accordingly.

B. Results

In this section, we will present the results of the evaluation of the Cyber-OF, and we will examine the impact of the following parameters:

- **End-to-end delay:** It is the duration between starting packet transmission and its reception by the DAG root.
- **Network lifetime** The network lifetime of a WSN is defined as the time collapsed until the first sensor runs out of energy.

1) Average delay: Fig 3 compares the average end-to-end delay of the three objective functions Energy-OF , Latency-OF and the Cyber-OF. It is obivious that they have similar delay values when the network is composed of less than 20 nodes. However , there is a slight difference for the energy OF which allows a higher average delay. This result is expected as the energy OF only maximizes the network lifetime and the choice of the best parent is based only on the energy remaining in the node. We notice also that when the network is composed of more than 20 nodes , the Cyber-OF experiences lower average delay values than the energy OF. This confirms the tendency of the Cyber-OF to minimize the delay when critical events (a fire alarm in our simulation) are detected.

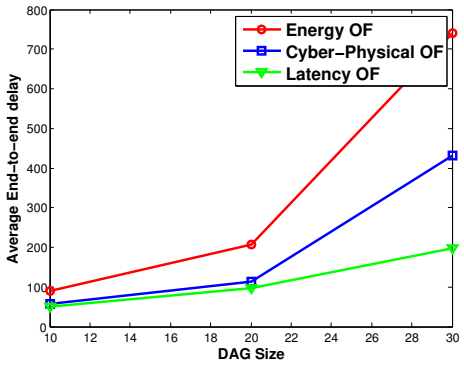


Fig. 3. Average End-to-End delay

2) Network lifetime: Figure 6 represents the energy consumed by latency OF and Cyber-OF and demonstrates how it can save energy and maximize the network lifetime more than the Latency-OF during 5 minutes of the simulation.

V. Conclusion and Future Work

In this paper, we presented the initial results and implementations of the cyber-physical objective function to adapt RPL to the properties of the environment. As a future work, the energy metric will be combined with other metrics in order to guarantee an acceptable QoS in the presence of a disaster or in normal conditions. We also aim at storing two parent candidates in the sensor to speed up the advertisement of the alarm. One is used when a critical event is detected and the other is used in normal conditions.

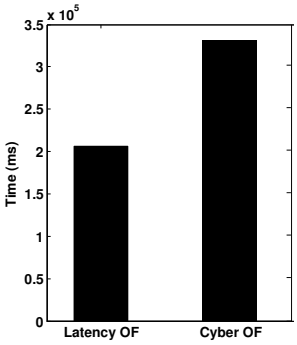


Fig. 4. Comparison between the network lifetime of the Latency-OF and the Cyber-Physical OF

VI. Acknowledgment

This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within the CISTER Research Unit (CEC/04234). And the author(s) would like to thank Prince Sultan University for funding this work through grant GP-CCIS-2013-11-10.

References

[1] K. L. Lueth, "IoT Market forecasts at a glance," 2014.
[2] Jonas, "Top 10 IoT applications: apple drives wearables to 1," 2015.
[3] O. Gaddour and A. Koubaa, "RPL in a nutshell: A survey," *Computer Networks*, vol. 56, no. 14, pp. 3163–3178, 2012.
[4] T. Winter, "Rpl: Ipv6 routing protocol for low-power and lossy networks," 2012.
[5] O. Gnawali, "The minimum rank with hysteresis objective function," 2012.
[6] O. Gaddour, A. Koubâa, and M. Abid, "Quality-of-service aware routing for static and mobile ipv6-based low-power and lossy sensor networks using RPL," *Ad Hoc Networks*, vol. 33, pp. 233–256, 2015.
[7] N. Gozuacik and S. Oktug, "Parent-aware routing for iot networks," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, pp. 23–33, Springer, 2015.
[8] H.-S. Kim, J. Paek, and S. Bahk, "Qu-rpl: Queue utilization based rpl for load balancing in large scale industrial applications," in *Sensing, Communication, and Networking (SECON), 2015 12th Annual IEEE International Conference on*, pp. 265–273, IEEE, 2015.
[9] P. Thubert, "Objective function zero for the routing protocol for low-power and lossy networks (rpl)," 2012.
[10] O. Gaddour, A. Kouba, R. Rangarajan, O. Cheikhrouhou, E. Tovar, and M. Abid, "Co-rpl: Rpl routing for mobile low power wireless sensor networks using corona mechanism," in *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, pp. 200–209, June 2014.
[11] O. Gaddour, A. Kouba, N. Baccour, and M. Abid, "Of-fl: Qos-aware fuzzy logic objective function for the rpl routing protocol," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2014 12th International Symposium on*, pp. 365–372, May 2014.

Inter-Arrival Curves for Multi-Mode and Online Anomaly Detection

Mahmoud Salem, Mark Crowley, and Sebastian Fischmeister
Department of Electrical and Computer Engineering
University of Waterloo, Canada
{m4salem, mcrowley, sfischme}@uwaterloo.ca

***Abstract*—Network Calculus uses arrival curves for requirement specification, which are extended in practice by Real-time Calculus for performance analysis and analytical modelling for arbitrary event streams. Recent work introduced *inter-arrival curves* as a specific form of arrival curves and demonstrated their feasibility as features to reason about the behavior of real-time systems for anomaly detection purposes. In this ongoing work, we present an application of inter-arrival curves in an online anomaly detection environment. We further employ inter-arrival curves in a multi-classifier framework for improved anomaly detection performance.**

I. Introduction & Related Work

Real-time systems often generate logs in a streaming fashion at a high-rate which makes traditional offline anomaly detection techniques ill-suited for online applications. One main reason is that the storage available by these systems is usually insufficient to accommodate a log of an entire session of operation. Our work aims to apply a technique that allows for online processing of trace streams. Figure 1 shows a proposed procedure for online mining of event traces. The online techniques require bounded computation time to analyze trace streams [1] due to the limited time available before new trace data arrives for processing. However, abnormal streams could be stored for further analysis offline, not requiring much storage compared to storing entire sessions of trace logging.

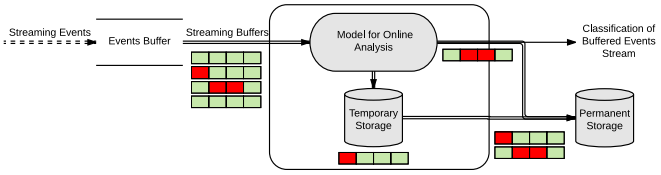


Fig. 1: Online Anomaly Detection Environment

The main target of anomaly detection techniques is to classify the behavior of the system under scrutiny either to be normal or anomalous. In this domain false negatives might lead to errors or failures in the system. False positives are also undesirable but merely lead to wasting resources investigating events which are not really anomalous. Thus, the aim of the classifiers is to maximize true positives while minimizing false positives [2]. In our recent work on offline anomaly detection [3], we employed a single classifier that analyzed event traces for a system having a single recurrent mode of operation. This technique, however, is not suited for systems having multiple modes of operations. Such systems may have intrinsic features that can vary considerably, yet still be operating normally. In this case, having a single classifier that learns the behavior of such systems can lead to higher false positives or negatives depending on the mode of operation. We aim to employ a multi-mode classification framework whose classifiers learn from the various normal modes of operations. Hence, the anomaly detection procedure can detect anomalous behavior within a given mode of operation more accurately.

The contributions of this work in progress are as follows:

- Propose an online anomaly detection technique for event traces using inter-arrival curves.
- Propose a multi-mode classification framework using inter-arrival curves for improved anomaly detection.
- Empirically demonstrate the feasibility and viability of the proposed approaches using event traces from an embedded real-time system.

II. Online Anomaly Detection

The presence of overwhelming data generated by real-time systems, combined with the need for real-time anomaly detection techniques, makes the problem of having an efficient online anomaly detection technique on the list of the current research problems [4].

Inter-arrival curves provide good features suited for anomaly detection [3]. Having features that provide information about system behavior with less data is a typical requirement for on-the-fly analysis specifically for real-time systems having recurrent behavior, that is why we conjecture that using inter-arrival curves restricts the online analysis to data generated within an instance of such recurrent behavior.

Informal problem statement: *Given a set of event traces generated by a well-specified system in a given execution scenario, check on-the-fly whether a stream of events from the same system originates from the same execution scenario.*

The first problem is to target online classification for trace streams. In prior inter-arrival work [3], we demonstrated an offline approach that classifies complete traces post-mortem. The online classification problem is less studied and faces several challenges [4]. One main challenge is the efficient classification of the trace stream constrained by both the computational approach and the trace length. Therefore, we split our problem into two sub-problems. The first sub-problem

is to implement the inter-arrival curves approach efficiently. The second sub-problem is to find a set of parameter settings within the inter-arrival curves framework [3] that would allow reasoning about the system behavior using a single-pass procedure on a minimal trace stream.

A. Efficient Curves Implementations

The computational approach of the online anomaly detection technique affects the analysis efficiency. For example, consider an online anomaly detector using a non-overlapping sliding window, the processing of an input stream should be at least as fast as the generation of data having a length equal to the defined window length. In other words, data within the sliding window should be processed before another window of data replaces it.

Formally, assume a data generation rate $D_r = \frac{|T_{ev}|}{time_D}$ where the $|T_{ev}|$ is the maximum number of events generated in a trace stream within $time_D$. In this case, the classifier should be able to reason about the stream in a time less than $time_D$, assuming the processed streams are not overlapping. The inter-arrival curves computation relies on a sliding window technique which has well studied, computationally efficient algorithms [5].

B. Parameters Suited for Online Analysis

Based on our prior offline anomaly detector [3], the primary parameter we need to choose for an online anomaly detection is Δ_{max}. Minimizing Δ_{max} in a way that achieves an acceptable ROC curve [2] would allow for lower computation time. Having a shorter window length relates to the desired $|T_{ev}|$ which, as we just demonstrated, is crucial for efficient computation.

Inter-arrival curves can be used to empirically compute a minimal Δ_{max} that achieves the acceptable ROC curve. During the offline training phase, the computation uses an upper bound of the Δ_{max} value where the validation phase can show classification results for the range of Δ values less than Δ_{max}. The online testing phase uses the minimal Δ that achieved the desired classification results during validation.

III. Multi-Mode Anomaly Detection

The second problem is to target improving classifier accuracy. The results obtained from the single-class classifier in [3] show that we could avoid false positives but we still could not detect all anomalies. We conjecture that some anomalies go undetected, because of the aggregation procedure implied by the single-classifier approach. In other words, a deviation of a curve can go undetected if it still complies with minimum and maximum counts obtained from different, yet normal curve.

We propose to experiment with classifiers tailored to the different modes of operation. We conjecture that such an approach would lower the number of false negatives. For example, consider a system having three modes of operation that represent the normal behavior of the system, yet having significantly different features. An anomaly in a mode of operation can be considered normal using an aggregate model of these significantly different features. However, the anomaly can be detected more accurately if we obtain a model using features that only describe a specific mode of operation during normal system behavior and vice versa.

Informal problem statement: *Given a set of event traces generated by a well-specified system that exhibits several modes of operations, check whether a new trace from the same system reflects any of these modes of operation.*

The problem statement studies a multi-class classification problem through modelling the normal behavior as multiple classes of behaviors where a class corresponds to one or more normal modes of operation. The classical challenge to this approach is labeling the classes. Ideally, such labels can exist within the trace as markers that indicate the different modes of operation. But in reality, a preprocessing phase can aim to detect the modes of operation and label the classes blindly without prior knowledge.

IV. Experimental Evaluation

A. Dataset

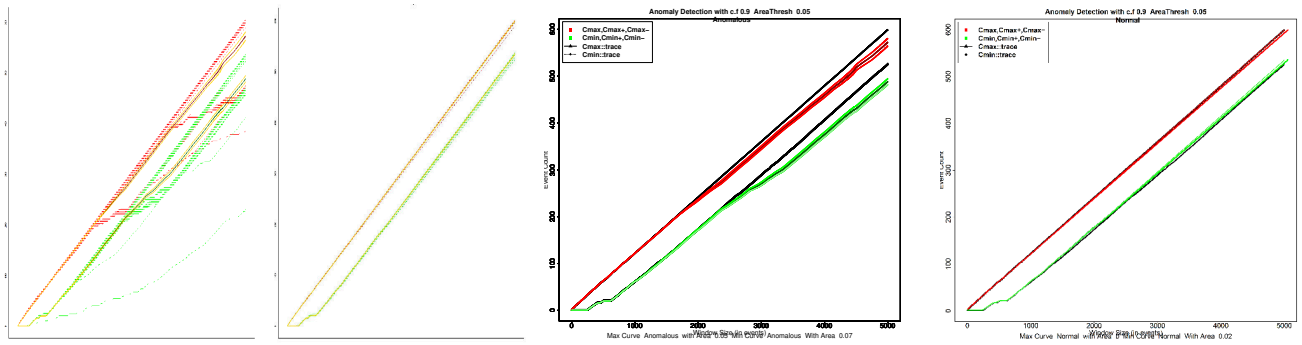
The device under test mimics data collection systems which periodically sense, process, and send data to a remote center. For the proposed preliminary work, we use markers that separate each mode of operation to validate our results. However, a more practical deployment would require inferring these markers automatically. The labels used in our experiments are as follows: 401-"Data Acquisition", 402-"Data Compression", 403-"Checksum", 404-"Communication", and 405-"Sleep". Anomalous behavior of the data collection system might include collecting larger chunks of data, changed storage medium, communication failures, etc. We use a BeagleBone board running QNX RTOS to collect the dataset. The traces contains QNX kernel events generated by the *tracelogger* utility according to the RTOS state machine [6]. For our experiments, an event is a combination of the *class, event* attributes. A trace for offline analysis contains $\simeq$ 25000 events while a trace for online analysis contains $\simeq$ 5000 events. Table II and Table I show the number of traces used in each experiment.

B. Preliminary Results

The preliminary evaluation of our approaches shows promising results following the conjectures discussed earlier.

Online Anomaly Detection The experiments use a synthetically streamed set of traces satisfying the criteria $|T| \simeq \Delta_{max}$ to evaluate the capability of the online approach. The approach uses a single classifier as in [3] that trains using traces of a given mode of operation, then during the testing phase the classifier aims to classify whether an online stream of traces originates from that mode of operation.

The classification results for online anomaly detection approach using inter-arrival curves are promising. During the testing phase, we choose smaller Δ_{max} and $|T|_{testing}$ for faster computation per stream, but enough to achieve acceptable classification results. During the offline training phase, we choose $|T|_{training} \gg |T|_{testing}$ to obtain a model descriptive



(a) Single-Mode Classifier Model (b) Multi-Mode Classifier Model (c) Single-Mode Classifier fails to detect Mode "402" (d) Multi-Mode Classifier detects Mode "402"

Fig. 2: Comparing Model and Classification for *THREAD-THRUNNING*

for the system behavior. In our preliminary experiments, the training model uses $|T| = 25000$ where the online testing phase uses $|T| \simeq 5000$ and $\Delta_{max} \simeq 1500$. We use an efficient Cartesian product implementation for the sliding window technique, e.g., an online processing of a trace having $|T| = 5000$ events takes ~ 0.5 seconds. We are currently considering faster implementations for on-the-fly computation.

Mode "401" Training	Mode "401" Testing	Mode "402" Testing	True Positive Rate	False Positive Rate
50 Files	129 Files	185 Files	84%	0%

TABLE I: Online Classification Results using $\Delta_{max} = 1500$

Multi-Mode Anomaly Detection The second set of experiments evaluates the capability of the offline multi-mode approach to detect anomalies within a mode of operation, i.e., detect the effect of an anomaly in a mode of operation against the normal behavior of this mode. The experiments inspect whether the online approach can classify traces better than our single-classifier technique [3]. The single classifier technique uses a training set having normal traces for all modes of operation, the testing aims to detect whether one mode of operation is experiencing an anomalous behavior. The multi-mode classifier learns a classifier for each mode of operation in the normal training set; During testing, each classifier contributes to the analysis of the anomalous trace.

Figure 2a and Figure 2b show the obtained models obtained for event *THREAD-THRUNNING*. The accuracy of the model degraded as shown in Figure 2a, because the scenarios aggregated are diverse and cover a wide range of event counts. However, a dedicated model for the "402-Data Compression" mode of operation can accurately represent the target mode of operation as shown in Figure 2d.

The anomalous behavior in our case is accessing a different number of data blocks through a different storage device. Such an anomaly should affect the events generated using the marker "402-Data Compression" due to having more blocks of data processed. All the other modes of operation remain the same. The aggregate model for the single classifier shows a model that uses the mean of counts originating from different behaviors; that is why the model fails to detect the anomaly with high accuracy. The same anomaly can be better detected using the obtained model for the classifier learned for this specific mode of operation, in this case the classification results from the classifiers learned for the other modes of operation can be discarded. Table II summarizes the classification results using both techniques. The tailored classifier in the multi-class environment shows a higher TPR and lower FPR compared to the single classifier environment, i.e., some files were incorrectly classified.

Normal Training	Mode "402" Normal Testing	Mode "402" Anomalous Testing	True Positive Rate	False Positive Rate
All Modes - 142 files	84 files	46 files	100%	42%
Mode "402" - 50 files	21 files	46 files	100%	0%

TABLE II: Single vs Multi-Mode Classifier Results

V. Conclusion

Following our recent work of using inter-arrival curves for anomaly detection using event traces [3], we study an improved classification approach that categorizes the different modes of operation for a given system. Also, we target the problem of the online classification where using inter-arrival curves shows a promising approach by having the capability of detecting anomalies using a window of events instead of an entire trace which makes it suited for on-the-fly analysis.

References

[1] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Online system problem detection by mining patterns of console logs," in *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE, 2009, pp. 588–597.

[2] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[3] M. Salem, M. Crowley, and S. Fischmeister, "Anomaly detection using inter-arrival curves for real-time systems," to appear in ECRTS, 2016.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection for Discrete Sequences: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 823–839, 2012.

[5] P. Esling and C. Agon, "Time-series data mining," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, p. 12, 2012.

[6] "Qnx state machine." [Online]. Available: http://www.qnx.com/developers/docs/am11/index.jsp?topic=%2Fcom.qnx.doc.neutrino.getting_started%2Ftopic%2Fs1_procs.html

Tightening worst-case timing analysis of Tilera-like NoC architectures

Hamdi Ayed, Jérôme Ermont, Jean-Luc Scharbarg and Christian Fraboul
Toulouse University - IRIT - ENSEEIHT
Email: {hamdi.ayed2, jerome.ermont, jean-luc.scharbarg, christian.fraboul}@enseeiht.fr

***Abstract*—Several approaches have been used for worst-case traversal time (WCTT) analysis of Tilera TILE64-like Network-on-Chip (NoC). Among them the recursive calculus (RC) simply captures the specific features of a NoC. Initial RC method is pessimistic since it does not take into account the implemented buffer capacity within routers and it does not integrate packet inter-arrival constraints. Approaches to overcome these limitations have been proposed. However, no existing work addresses both limitations at the same time. The goal of this paper is to bridge this gap by combining existing approaches.**

I. Introduction

Many-core architectures are promising candidates for hard real-time systems. They are based on simple cores interconnected by a Network-on-Chip (NoC). Timing constraints, such as bounded delays, have to be guaranteed for hard real-time systems. However, the initial motivation when designing NoCs was to increase the average case throughput. NoCs can thus be used in hard real-time systems either by analyzing the Worst-Case Traversal Time (WCTT) of flows on existing many-cores or by modifying the hardware so that no contentions can occur by design, leading to straightforward WCTT for flows.

NoCs based on the second approach [7], [8], [9] are not available in commercially existing many-core architectures, such as the STMicroelectronics P2012/STHORM fabric [11], the Tilera Tile CPUs [12] or the KalRay MPPA [4].

In this work, we focus on Tilera Tile64 [12]. It is composed of a set of routers organized in 2D-mesh structure as depicted in Figure 1. Links are full-duplex. Flows with static paths are transmitted on this architecture. We denote by $path(f)$ the ordered list of links for flow f from its source to its destination. Flow control is implemented in each router: a packet cannot be forwarded if the next output port on its path is busy. It corresponds to a classical wormhole switching mechanism [10]. A packet is divided in flow control digits (flits) of fixed size. The first flit contains routing information. In each cycle one flit is forwarded from each router to the following one, provided that its destination output port is not busy. Each router includes five full duplex input-output ports. A three flits buffer is associated to each input port. Input ports are polled, based on Round-Robin Arbitration (RRA). The first packet in the buffer of the polled input port is forwarded flit by flit to the next router on its path if the buffer of the corresponding input port is not full. Otherwise the next input port (based on RRA) is polled.

This strategy requires very small buffers in routers. However it generates indirect blocking, as illustrated in Figure 1. In this example, flow f_1 can be blocked by flow f_3 in router R_2. Since f_3 can be blocked by f_4 later on its path, f_1 can be indirectly blocked by f_4.

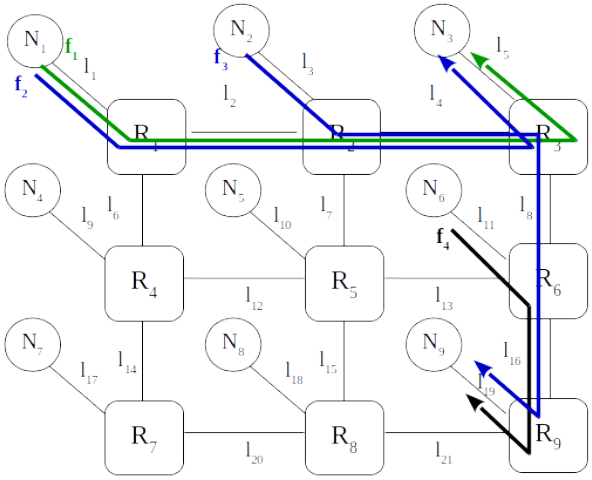


Fig. 1. Case study

Several approaches have been proposed for Worst-Case-Traversal-Time (WCTT) computation of Tilera-like NoC [5]. Among them, the Recursive Calculus (RC) method offers a simple way to capture the wormhole effect with direct and indirect blocking. Sources of pessimism have been identified in the initial RC method.

First, the analysis ignores buffer size. Thus it considers that a flow f_i is blocked by another flow f_j until f_j fully reaches its destination. It has been shown that considering this buffer effect leads to tighter WCTT [2].

Second, the analysis does not take into account flow periodicity. Thus, it considers that more than one frame of a flow f_j can delay a flow f_i, "even though this is impossible considering f_j's period. The RC approach has been extended to integrate this periodicity constraint [3].

The contribution of this paper is to propose an enhanced RC method integrating both buffer effect and flow periodicity.

II. Extended WCTT approach

The best case traversal time for a packet of a flow f occurs when the packet experiences no contention in routers. In this specific case, the traversal time is:

$$d_{min}(f) = n_h(f)*d_c+(n_h(f)-1)*d_{sw}+(S(f)-1)*d_c \quad (1)$$

$n_h(f)$ represents the number of links in $path(f)$, d_c is the time needed to transmit a flit on a link, while d_{sw} is the switch latency. Thus, $n_h * d_c + (n_h - 1) * d_{sw}$ is the time needed to transmit the first flit from source to destination. The packet includes $S(f)$ flits. Thus, it takes $(S(f) - 1) * d_c$ to transmit the remaining flits (pipeline transmission at flit level). In this paper, we assume that $d_c = d_{sw} = 1\ cycle$, which means that a flit requires exactly one cycle to be forwarded from one node to the following, and one cycle to be processed in a router.

A flow might be delayed by other flows. The worst-case traversal time (WCTT) of a flow f corresponds to the largest possible impact of other flows on f delay.

In [6], the authors have formalized the recursive calculus method which simply integrates all possible blocking times induced by direct and indirect flows. For each flow f_i, a list of blocking packets is constructed by following the path of f_i and identifying the packets blocking it directly, and the packets directly blocking these direct blocking packets, and so on. For f_1 in the example in Figure 1:

- f_1 can be delayed by the transmission of a packet from f_2 at its source node N_1: $Seq(f_1) = \{f_2, f_1\}$;
- f_2 can be delayed by a packet from f_3 at router R_2: $Seq(f_1) = \{f_3, f_2, f_1\}$;
- f_3 can be delayed by the transmission of a packet from f_4 at router R_6: $Seq(f_1) = \{f_4, f_3, f_2, f_1\}$;
- f_1 can be delayed by another occurrence of f_3 at R_2: $Seq(f_1) = \{f_4, f_3, f_2, f_3, f_1\}$;
- f_3 can be delayed by another occurrence of f_4 at R_6: $Seq(f_1) = \{f_4, f_3, f_2, f_4, f_3, f_1\}$;

Thus, we get: $Seq(f_1) = \{2 * f_4, 2 * f_3, f_2, f_1\}$.

This sequence represents the blocking time contributions of all the flows that may directly or indirectly block f_1. f_1 is the last element of the list and it is delayed by all the preceding packets in $Seq(f_1)$. Then, the resulting WCTT bound is computed:

$$WCTT(f_1) = 2{*}d(f_4, f_1){+}2{*}d(f_3, f_1){+}d(f_2, f_1){+}d_{min}(f_1)$$

where, $d(f_j, f_i)$ is the transmission time of f_j starting from the first common node with f_i to its destination. For example:

- $d(f_2, f_1) = 4 * d_c + 3 * d_{sw} + (S(f_2) - 1) * d_c$

With three flit packets, $WCTT(f_1) = 50\ cycles$.

This computation assumes that f_2, f_3 and f_4 packets block f_1 till they reach their destination. This assumption simplifies the computation. However it might introduce some pessimism.

As an illustration, let's consider three flit packet size and a three flit buffer in each input port. A packet from f_3 can be fully stored in R_6 input buffer. Consequently, the impact of an f_3 packet on both f_1 and f_2 stops as soon as it leaves R_3. Since it can leave R_3 even if there is a pending packet from f_4, f_4 doesn't add any extra delay for f_1 and f_2. Thus the worst-case sequence of packets blocking f_1 becomes $Seq(f_1) = \{2 * f_3, f_2, f_1\}$. This drops down $WCTT(f_1)$ to $40\ cycles$.

This first source of pessimism is addressed in [2]. The authors consider one flit buffers. They show that, in the situation where a flow f is directly blocked by a flow f_d, which is directly blocked by a flow f_{id} later on its path, f_{id} cannot influence f if the number of routers between the point where f_d leaves f and the point where f_d meets f_{id} is at least f_d packet size. On the example in Figure 1, it comes to say that, if f_3 packet size is one, f_4 has no impact on f_1 or f_2 (one flit input buffers are assumed in [2]).

This property can be generalized for B flit input buffers. The number of routers needed to store an f_d packet is

$$N_{f_d}^B = \left\lceil \frac{S(f_d)}{B} \right\rceil$$

If f_d is blocked by f_{id} after passing $N_{f_d}^B$ routers from the point where it leaves f, then f_{id} has no influence on f.

Considering the example in Figure we have $N_{f_3}^3 = \lceil \frac{3}{3} \rceil = 1$. Since f_3 is blocked by f_4 after passing one router from the point where it leaves f_1, f_4 has no influence on f_1.

A second source of pessimism in the initial RC computation is due to the fact that flows are sporadic. It means that there is a minimum duration T_j between the generation times of two consecutive packets from a flow f_j. Thus there is a minimum duration $T_j - J_j^{R_l}$ between arrival times of two consecutive packets from f_j in router R_l, where

$$J_j^{R_l} = WCTT(f_j, R_l) - d_{min}(f_j, R_l)$$

i.e. the difference between the worst-case delay for f_j from its source to R_l and its best-case delay. Then the maximum number of packets from f_j which can arrive in router R_l, in a time window of t cycles, is bounded by:

$$Nb(f_j, R_l, t) = \left\lceil \frac{t}{T_j - J_j^{R_l}} \right\rceil \quad (2)$$

This constraint on the maximum number of packets from a flow in an interval is not considered in the initial RC computation. Thus the computed sequence might contain too many packets from a given flow.

Coming back to the example in Figure 1, the computation considers that f_1 can be delayed by two packets from f_3 (as well as two packets from f_4). Such a scenario is possible only for small enough inter-arrival time constraints between packets from f_3 (as well as packets from f_4). As an example, let's consider an inter-arrival time of at least 100 cycles between two consecutive packets from f_3 (and f_4). Since packets from f_3 cannot be blocked before router R_2, they experience no jitter ($J_3^{R_2} = 0$). We have shown that the worst-case traversal time of f_1 cannot be more than 50 cycles (based on the RC appoarch in [5]). It gives an overestimation of the interval of time in which packets from f_3 have to arrive in R_2 in order to delay f_1. Then an upper bound on the number of packets from f_3 which can delay f_1 is given by:

$$Nb(f_3, R_2, 50) = \left\lceil \frac{50}{100 - 0} \right\rceil = 1 \quad (3)$$

Similarly, only one frame from f_4 can delay f_1. Therefore, $Seq(f_1)$ can be $Seq(f_1) = \{f_4, f_3, f_2, f_1\}$. This leads to $WCTT(f_1) = 36\ cycles$.

This second source of pessimism is addressed in [3]. The basic idea consists in enumerating all possible sequences of

blocking for a given flow, respecting the minimum inter-release time constraint. For flow f_1 of the example in Figure 1, it leads to the following list of sequences:

$$\{\{f_1\},\{f_2,f_1\},\{f_3,f_2,f_1\},\{f_2,f_3,f_1\},\{f_4,f_3,f_2,f_1\},\{f_2,f_4,f_3,f_1\}\}$$

We propose to consider an overestimation of all enumerated sequences in order to mitigate this explosion problem: the initial sequence (obtained by RC approach without inter-arrival constraints) leads to a WCTT for the flow under study, for each interfering flow, we compute the maximum number of packets within this WCTT. This number might be unreachable since it assumes that each flow can have one packet at the beginning as well as at the end of the sequence, which might be impossible. However, it is a safe overestimation of the number of interfering packets.

The proposed approach in this paper can be summarized as follows:

- Sequence computation based on the initial RC approach;
- Elimination of non influencing indirect blocking flows, thanks to buffer effect;
- Elimination of packet occurrences, based on inter-arrival constraints.

Based on this approach, we get $Seq(f_1) = \{f_3, f_2, f_1\}$, leading to $WCTT(f_1) = 31\ cycles$.

III. Experiments

We consider the real-time application described in Figure 2, where 10 flows need to be exchanged on a 6*6 2D-mesh NoC. Flows are periodic and have static paths. We assume a packet size of 10 flits and a period of 500 cycles for all flows.

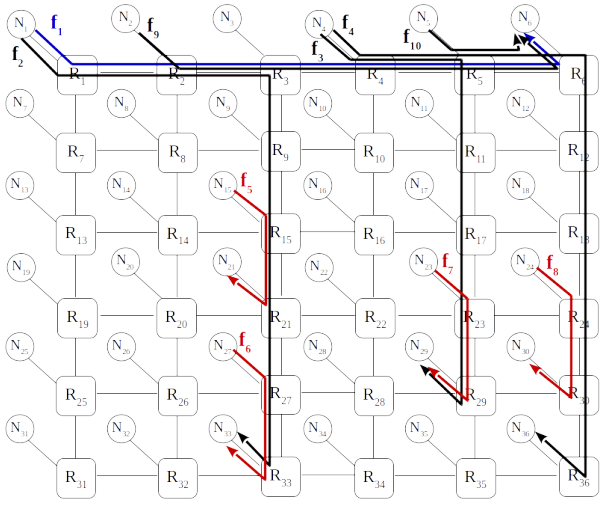


Fig. 2. Case study

In Table I, we reported the WCTT bounds obtained using respectively: (i) the initial RC method [5]; (ii) extended RC algorithm based on work in [1]; (iii) extended RC algorithm in [3]; (iv) our extended algorithm, as described in the previous section. Results are shown in Table I. Our approach gives tighter WCTT bounds for all the flows.

TABLE I
Bounds on WCTT

Flow	Initial RC [5] (cycles)	[1] bounds (cycles)	[3] bounds (cycles)	Extended RC (cycles)
f_1	304	251	121	111
f_2	304	264	121	98
f_3	143	112	86	78
f_4	143	127	86	79
f_5	44	40	27	25
f_6	24	24	23	22
f_7	27	25	25	23
f_8	24	24	23	22
f_9	167	145	89	82
f_{10}	50	44	30	27

IV. Conclusion

We propose in this paper an enhanced RC method for worst-case traversal time analysis of Tilera TILE64-like Network-on-Chip (NoC). The proposed enhanced method integrates both buffer effect and flow periodicity. It significantly improves WCTT analysis on our case study. This approach still has to be formalized and applied on larger industrial case study. It should also be extended to other kinds of NoCs.

References

[1] L. Abdallah, M. Jan, J. Ermont, and C. Fraboul. Optimizing worst case delay analysis on wormhole networks by modeling the pipeline behaviour. In *Proc. of the 13th Intl. Workshop on Real-Time Networks (RTN'2014)*, Madrid, Spain, July 2014.

[2] L. Abdallah, M. Jan, J. Ermont, and C. Fraboul. Wormhole networks properties and their use for optimizing worst case delay analysis of many-cores. In *Industrial Embedded Systems (SIES), 2015 10th IEEE International Symposium on*, pages 1–10, June 2015.

[3] D. Dasari, B. Nikolić, V. Nélis, and S. M. Petters. NoC Contention Analysis Using a Branch-and-prune Algorithm. *ACM Trans. Embed. Comput. Syst.*, 13(3s):113:1–113:26, Mar. 2014.

[4] B. D. de Dinechin, D. van Amstel, M. Poulhiès, and G. Lager. Time-critical computing on a single-chip massively parallel processor. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, March 24-28, 2014*, pages 1–6, 2014.

[5] T. Ferrandiz, F. Frances, and C. Fraboul. A method of computation for worst-case delay analysis on SpaceWire networks. In *Proc. of the 4th Intl. Symp. on Industrial Embedded Systems (SIES)*, pages 19–27, Lausanne, Switzerland, July 2009.

[6] T. Ferrandiz, F. Frances, and C. Fraboul. A Network Calculus Model for SpaceWire Networks. Proceedings of the EEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2011), 2011.

[7] K. Goossens, J. Dielissen, and A. Radulescu. Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 22(5):414–421, 2005.

[8] A. Hansson, M. Subburaman, and K. Goossens. Aelite: A flit-synchronous network on chip with composable and predictable services. In *Proc. of the Conf. on Design, Automation and Test in Europe (DATE'09)*, pages 250–255, Nice, France, 2009.

[9] S. M. G. S. Kasapaki, Schoeberl. Argo: A real-time network-on-chip architecture with an efficient gals implementation. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 479–492, 2016.

[10] L. Ni and P. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Transactions on Computers*, 26(2):62–76, Feb 1993.

[11] D. Rahmati, S. Murali, L. Benini, F. Angiolini, G. De Micheli, and H. Sarbazi-Azad. Computing accurate performance bounds for best effort networks-on-chip. *IEEE Transactions on Computers*, 62(3):452–467, March 2013.

[12] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. B. III, and A. Agarwal. On-chip interconnection architecture of the tile processor. 2007.

Mixed-criticality scheduling with memory regulation

Muhammed Ali Awan*, Konstantinos Bletsas*, Pedro Souto†*, Benny Akesson*, Eduardo Tovar*, Jibran Ali*
*CISTER/INESC-TEC, ISEP/IPP, Portugal †Faculty of Engineering, University of Porto, Portugal

***Abstract*—The state-of-the-art models and schedulability analysis for mixed-criticality multicore systems overlook low-level aspects of the system. To improve their credibility, we therefore incorprate, in this work, the effects of delays from memory contention on a shared bus. Specifically, to that end, we adopt the predictable memory reservation mechanism proposed by the Single Core Equivalence framework. Additionally, we explore how the reclamation, for higher-criticality tasks, of cache resources allocated to lower-criticality tasks, whenever there is a criticality (mode) change in the system, can improve schedulability.**

I. Introduction

On mixed-criticality scheduling: The integration of functionalities of different criticalities on a multicore necessitates some scheduling isolation between criticality levels. The most notable analytical model for those problems, by Vestal [1], in its basic form, assumes two criticality levels, high and low. High-criticality tasks (H-tasks) have two different estimates of their worst-case execution time (WCET): The L-WCET which is *de facto* deemed safe, and the H-WCET, which is *provably* safe and possibly much greater. For low-criticality tasks, only the L-WCET is defined. There are two modes of operation. The system boots and remains in low-criticality mode (L-mode) as long as no job (instance of a task) executes for longer than its L-WCET. However, if any job exceeds its L-WCET then the system immediately switches into high-criticality mode (H-mode) and drops all L-tasks. It is pessimistically assumed that in H-mode all jobs by H-tasks (including any existing jobs at the time of the mode switch) may execute for up to their H-WCET. Under these assumptions, it must be provable offline that (i) no task misses a deadline in L-mode and (ii) no H-task misses a deadline in H-mode.

Various scheduling approaches for this model exist. Below, we only discuss works employing *deadline-scaling*. This technique originated with the EDF-VD (Earliest Deadline First - with Virtual Deadlines) scheduling algorithm [2] for implicit-deadline mixed-criticality sporadic task sets. EDF-VD uses standard EDF scheduling rules but instead of reporting the real task deadlines to the scheduler, for the purpose of scheduling decisions, it reports shorter deadlines (if needed) for H-tasks during L-mode operation. In doing so, it prioritises H-tasks more than conventional EDF would, over parts of the schedule. This allows H-tasks to be sufficiently ahead of schedule such that they can catch up with their true deadlines if any task overruns its L-WCET. After the switch to H-mode, the true H-task deadlines are used for scheduling and L-tasks are dropped.

This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within the CISTER Research Unit (CEC/04234); also by by FCT/MEC and the EU ARTEMIS JU within project ARTEMIS/0001/2013 - JU grant nr. 621429 (EMC2).

EDF-VD proportionately scales all H-task deadlines by a common factor. Ekberg and Yi improve on this by using distinct scaling factors for different H-tasks and a more precise demand bound function (dbf) based schedulability test [3].

Access isolation for caches, memory buses and DRAMs: A prominent approach for making multicores more predictable regarding sources of contention is Single-Core Equivalence (SCE) [4]. Under SCE, fixed-priority scheduling is used and MemGuard [5], a periodic software mechanism, regulates memory accesses from different cores. Over a fixed interval called the *regulation period*, all cores get an equal "slice" of the overall memory bandwidth. This assumes that all memory accesses go through the same memory controller. MemGuard stalls any core that exceeds its share, until the start of the next regulation period. The analysis must consider such *regulation stalls* in addition to conventional *contention stalls*, caused by contention between different cores at the DRAM controller.

SCE's stall-aware schedulability analysis [6], characterises each task by its *WCET in isolation* and its worst-case number of *residual memory accesses*. These correspond to the WCET when no other task is present in the system and an upper bound on the number of memory accesses by the task that go all the way to DRAM. It is also assumed that each task has its most frequently accessed pages locked in place in the shared last-level cache by the Colored Lockdown [7] mechanism. This arrangement promotes determinism by eliminating inter-task interference in the cache and Mancuso et al. use this information to more tightly characterise the WCET in isolation and the number of residual memory accesses. Both quantities decrease as the number of locked pages increases [6]. Using these derived attributes for each task, Mancuso et al. then calculate contention stall and regulation stall terms for the tasks that add to their response time, assuming round-robin memory arbitration. Their analysis also assumes *DRAM Bank Partitioning* via the OS-level memory allocator PALLOC [8].

Contribution #1: We combine Ekberg and Yi's work and SCE. The rationale is that the former overlooks the effects of contention for platform resources. Conversely, SCE provides a form of scheduling isolation over those resources that is handy for safety-critical applications, but is criticality-oblivious.

Contribution #2: We identify and leverage an opportunity for improved schedulability and resource efficiency, afforded by such a unified approach. Consider a system conforming to the SCE principles (including the cache partitioning) and scheduled by a variant of Ekberg and Yi's algorithm. When a mode change occurs, L-tasks are dropped, which means that their cache partitions are no longer of any use. We therefore assign them to the remaining H-tasks, after the mode change,

to accommodate additional pages by the latter. This re-uses resources that would otherwise be idle and lowers the H-WCETs of jobs released after the mode change, leading to better scheduling performance. The challenge then is to develop schedulability analysis that safely quantifies the improvement.

II. System model and assumptions

We assume m identical physical cores (π_1 .. π_m) accessing DRAM via a shared memory controller. A core can have many outstanding memory requests. Prefetchers/speculative units are disabled. Our assumptions are inline with SCE [4]:

- The last-level cache is shared among the cores. The Colored Lockdown mechanism [7] is used to mitigate the intra-/inter-core interference. It allows a task to lock its most frequently used pages in the last-level cache, providing a deterministic approach to compute the residual number of memory accesses and WCET as a function of the number of locked pages.
- DRAM-bank partitioning (e.g., using PALLOC [8]) is not strictly required. However, lower and upper bounds on the memory access time of a single transaction (L_{min} and L_{max}, respectively) are needed as inputs. These can be computed as in [6]. The DRAM controller is round-robin.
- The memory bandwidth is managed via the MemGuard [5] OS-level regulation mechanism. It uses the the worst-case memory access time L_{max} to determine the maximum number of memory accesses, K, during a regulation period P, as $K \stackrel{\text{def}}{=} \frac{P}{L_{max}}$. The memory bandwidth of K memory accesses is distributed equally among different cores, i.e. $K_i \stackrel{\text{def}}{=} \frac{K}{m}$ requests per regulation period for core π_i. This budget is allocated at the start of the regulation period. The execution of a core is stalled until the end of current regulation period as soon as it exhausts its per-regulation-period budget of K_i memory requests.

Each task τ_i has a minimum inter-arrival time T_i, a relative deadline D_i (with $D_i \leq T_i$) and a criticality level $\kappa_i \in \{L, H\}$ (low or high, respectively). The subsets of low-criticality and high-criticality tasks are defined as $\tau(L) \stackrel{\text{def}}{=} \{\tau_i \in \tau | \kappa_i \in L\}$ and $\tau(H) \stackrel{\text{def}}{=} \{\tau_i \in \tau | \kappa_i \in H\}$. However, since (i) the (actual) WCET of a task depends on its number of pages (selected in order of access frequency) locked in place in the last-level cache and (ii) different *estimates* of that WCET (derived via different techniques), are to be used for the L-mode and H-mode, we extend Vestal's model by assuming that for each task, a different L-WCET/H-WCET pair can be available as input, for each possible value of the number of locked "hot" pages. For example $C_i^L(6)$ denotes the L-WCET of τ_i when this task is configured with its 6 "hottest" pages locked in the cache. In detail:

The Colored Lockdown approach determines the most frequently accessed (hot) pages for each task through the profiling framework in [7]. The WCET of a task in isolation is computed as a function of the number of hot pages locked in the last-level cache and represented as a *progressive lockdown curve* (WCET vs locked pages in last-level cache). The increase in number of locked pages in the last-level cache decreases the last-level cache misses and, consequently, also the WCET of the task. The approach proposed by Mancuso et al. [7] is a measurement-based technique, so its outputs are not provably safe, but they can serve as L-WCETs. However, some static analysis tools comprehensively cover all possible control flows (or even some infeasible paths) in a task, and these can be used for estimating the H-WCETs.

By safely modelling accesses to the hot pages locked-in by Colored Lockdown as "always hit", the static analysis tool can derive tighter WCET estimates than it would without this knowledge – and the improvement will be greater the more pages are locked in the cache. Hence, a progressive lockdown curve similarly exists for the H-WCET $C_i^H(\cdot)$.

Similarly to the WCETs, we extend the use of techniques of different conservativeness for estimating the residual number of memory accesses (last-level cache misses) of each task in L-mode and H-mode. Similar to WCET estimates, these values are functions of locked pages in the last-level cache ($\mu_i^L(\cdot)$, $\mu_i^H(\cdot)$) that decrease with the number of locked pages in the last-level cache. The utilisation of a task in L-mode (H-mode) of operation is defined as $U_i^L(\sigma) \stackrel{\text{def}}{=} \frac{C_i^L(\sigma)}{T_i}$ (resp., $U_i^H(\sigma) \stackrel{\text{def}}{=} \frac{C_i^H(\sigma)}{T_i}$), for σ locked pages in the last-level cache. We assume fully partitioned scheduling. i.e, no task ever migrates.

III. Cache scaling for mixed criticality system

A. Memory regulation stall

The MemGuard regulation mechanism bounds the number of DRAM accesses by each core. A core can be stalled for two main reasons. MemGuard allocates a memory access budget to each core per regulation period P. A core exceeding its budget is stalled until the start of the next regulation period; this is a *regulation stall*. As all cores share the DRAM controller, any delay in serving the memory accesses of a core due to accesses by other cores is a *contention stall*.

Mancuso et al [6] assume an even allocation of memory access budgets among cores and round-robin memory bus sharing. With these assumptions, the worst-case regulation-induced stall is computed for a given task τ_i performing $\mu_i(\cdot)$ residual memory accesses. It is shown that the regulation stall always dominates the contention stall in a given regulation period P. The duration of such a stall is $(P - K_i L_{min})$. The contention stall for any number of memory accesses $K_q \leq K_i$ within a given regulation period P is equal to $(m-1)K_q L_{max}$ under round robin arbitration. It has been shown [6, Th. 1]) that the stall due to $\mu_i(\cdot)$ memory requests issued by a task τ_i can be computed as:

$$\begin{aligned} stall(\tau_i, c, \sigma) = &\left\lceil \frac{\mu_i^c(\sigma)}{K_i} \right\rceil (P - K_i L_{min}) \\ &+ (m-1)(\mu_i^c(\sigma)) - (\left\lceil \frac{\mu_i^c(\sigma)}{K_i} \right\rceil - 1) K_i) L_{max} \end{aligned} \quad (1)$$

where, $c \in \{L, H\}$ and σ is the number of locked pages.

The L-WCET and H-WCET of a task τ_i with interference from other cores on the shared bus with σ locked pages in the last-level cache are defined as follows.

$$\bar{C}_i^L(\sigma) \stackrel{\text{def}}{=} C_i^L(\sigma) + stall(\tau_i, L, \sigma) \quad (2)$$

$$\bar{C}_i^H(\sigma) \stackrel{\text{def}}{=} C_i^H(\sigma) + stall(\tau_i, H, \sigma) \quad (3)$$

These values are computed using Equations (1)–(3), $\forall \sigma \in \{0, \ldots, \sigma_T\}$ on the progressive lockdown curves derived for the L-mode and H-mode of operation.

B. Last-level cache allocation

Let σ_i^L and σ_i^H be the number of pages by τ_i in the last-level cache in the two modes and σ_T be the total number of pages that fit in that cache. A good heuristic could be to set the σ_i^L values such that the total task set utilisation in L-mode is minimised. Intuitively, lower utilisation correlates with better schedulability. An optimal way to solve this heuristic would be the following Integer Linear Programming model (ILP):

$$\text{Mimimise} \sum_{\forall \tau_i \in \tau} \frac{\bar{C}_i^L(\sigma_i^L)}{T_i}, \text{ subject to} \sum_{\forall \tau_i \in \tau} \sigma_i^L \leq \sigma_T \quad (4)$$

Next, without backtracking, we determine the σ_i^H values.

$$\text{Mimimise} \sum_{\forall \tau_i \in \tau(H)} \frac{\bar{C}_i^H(\sigma_i^L)}{T_i} \quad (5)$$

$$\text{s.t. } \sigma_i^H \geq \sigma_i^L, \forall \tau_i \in \tau(H) \text{ and } \sum_{\forall \tau_i \in \tau(H)} \sigma_i^H \leq \sigma_T \quad (6)$$

The constraint in Equation (4) ensures that the total number of allocated pages does not exceed the available capacity of the cache. Pages abandoned by the L-tasks in the H-mode are "recycled" for H-tasks by the constraints in Equation (6). One counter-intuitive property of our proposed heuristics is that there may be a scenario in which the $\bar{C}_i^H(\sigma_i^H) \leq \bar{C}_i^L(\sigma_i^L)$. This could happen if the effects of the reduction of residual memory accesses for H-tasks in H-mode, from the additional pages, offset the pessimism from using a more conservative estimation technique for H-WCETs than for L-WCETs. Therefore, unlike the "classic" Vestal's model, we may have $\bar{C}_i^H(\sigma_i^H) \leq \bar{C}_i^L(\sigma_i^L)$ and better schedulability in the H-mode when compared to Ekberg and Yi's analysis.

C. Schedulability analysis

In the last step, we analyse the intra-core interference that a task may have due to memory accesses by other tasks running on the same core and integrate it to the schedualability analysis. In this paper, we have adapted Ekberg and Yi's schedulability analysis [3] to our system model. Ekberg and Yi proposed a demand-bound-function (dbf) based analysis that assumes tasks are scheduled with Earliest Deadline First (EDF). The deadlines of H-tasks are shortened in L-mode, so that they can stay "ahead of schedule" and perform a smooth transition from L- to H-mode of operation. In our approach, we currently perform no deadline-scaling but get a similar effect by allocating extra pages to the H-tasks in the last-level cache in H-mode[1]. Since Ekberg and Yi's technique is still usable as a schedulability test when the scaling factors are given as input, we use a scaling factor of 1 (no scaling) for all tasks. As in that work, we check for schedulability in the L-mode with a dbf-based [9] test. These derivations do not consider the fact that a task may be preempted by other tasks allocated to the same core. With the memory regulation employed, each preempting job in the worst-case may exhaust the allocated memory access budget within a given period P and hence, may be causing the core to stall upon the resumption of the preempted job. A job cannot preempt another job more than once. Therefore, one approach is to include the preemption overheads into the budget of preempting task.

Similar considerations apply to the adaptation of the test for cheching the schedulability in H-mode. Note though that any H-jobs caught up in the mode change may execute for up to $\bar{C}_i^H(\sigma_i^L)$ time units, since only subsequent H-jobs will execute with more hot pages (σ_i^H) in the cache.

IV. Conclusions

Using low-level information about the hardware platform and access regulation mechanisms, has potential to improve schedulability and confidence in the analysis of mixed-criticality systems. This work-in-progress paper is one step in that direction. Currently, we are developing a mixed criticality schedulability test that incorporates intra-core interference while ensuring the schedulability in L-mode, H-mode and the transition phase. Next, we will develop a testbed that incorporates these details and provides a platform to compare its performance with state-of-the-art mixed criticality scheduling algorithms. We hope this practical approach will outperform the existing state-of-the-art solutions.

References

[1] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in 28th RTSS, 2007.

[2] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in 24th ECRTS, July 2012, pp. 145–154.

[3] P. Ekberg and W. Yi, "Bounding and shaping the demand of mixed-criticality sporadic tasks," in 24th ECRTS, July 2012, pp. 135–144.

[4] L. Sha, M. Caccamo, R. Mancuso, J.-E. Kim, M.-K. Yoon, R. Pellizzoni, H. Yun, R. Kegley, D. Perlman, G. Arundale et al., "Single core equivalent virtual machines for hard realtime computing on multicore processors," Univ. of Illinois at Urbana Champaign, Tech. Rep., 2014.

[5] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "Memory bandwidth management for efficient performance isolation in multi-core platforms," Trans. Computers, vol. 65, no. 2, pp. 562–576, Feb 2016.

[6] R. Mancuso, R. Pellizzoni, M. Caccamo, L. Sha, and H. Yun, "WCET(m) estimation in multi-core systems using single core equivalence," in 27th ECRTS, July 2015, pp. 174–183.

[7] R. Mancuso, R. Dudko, E. Betti, M. Cesati, M. Caccamo, and R. Pellizzoni, "Real-time cache management framework for multi-core architectures," in 19th RTAS, 2013, pp. 45–54.

[8] H. Yun, R. Mancuso, Z.-P. Wu, and R. Pellizzoni, "PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms," in 20th RTAS, April 2014, pp. 155–166.

[9] S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," J. Real–Time Syst., 1990.

[1] In time, we intend to also incorporate (and speed up) the deadline-scaling.