


RTNS 2010

18th International Conference on Real-Time and Network Systems

Actes JRWRTC'10

Workshop rtns

4-5 November 2010

www.irit.fr/rtns2010

TOULOUSE

SPONSORS

IEEE France Section

INP Toulouse

IRIT CNRS - INPT - UPS - UT1 - UTM

Université de Toulouse

Université Paul Sabatier Toulouse III

Université Toulouse 1 Capitole

Conception : L.CHACUN - IRIT - 05 61 55 76 90

 - ISBN : 978-2-917490-11-2 - EAN : 9782917490112


Foreword

This volume contains the papers presented in the 4rd Junior Researcher Workshop on Real-Time Computing (JRWRTC'10), held in conjunction with the 18th International Conference on Real Time and Network Systems (RTNS'10) on November 4-5 in Toulouse, France. As with the former editions, the goal of the workshop is to bring together junior researchers – mostly PhD students and post-docs – who work on real-time issues so they can present and exchange their ideas and more generally review current trends of the domain. Through a short presentation (including posters), the participants will be able to discuss their current work with all conference attendees.

We received around ten submissions, each of them were reviewed by three members of the program committee. The papers encompass various themes, from Worst-Case Execution Time (WCET) to Quality of Service (QoS) analysis, from specification and modeling to verification, etc. The committee decided to accept seven of them.

We want to congratulate authors for the quality of their submissions to JRWRTC'10. The programme committee members have also made a great job in a limited amount of time, and we would definitely recommend the work of any of them.

October 2010

Vandy Berten
Roman Bourgade
Xiaoting Li

Workshop Organization

Program Chairs

Vandy Berten, ULB, Bruxelles, Belgium
Roman Bourgade, IRIT, Toulouse, France
Xiaoting Li, IRIT-ENSEEIHT, Toulouse, France

Program Committee

Sebastian Altmeyer, Universität des Saarlandes, Germany
Moris Behnam, Mälardalen University, Sweden
Björn B. Brandenburg, University of North Carolina, USA
Christian Fotsing, LISI, Poitiers, France
Damien Hardy, IRISA, Rennes, France
Pi-Cheng Hsiu, Academia Sinica, Taiwan
Osmar Marchi dos Santos, University of York, UK
Mohamed Marouf, INRIA Rocquencourt, France
Patrick Meumeu Yomsi, ULB, Bruxelles, Belgium
Aurelien Monot, LORIA, Nancy, France
Vincent Nélis, ULB, Bruxelles, Belgium
Marco Paolieri, BSC, Spain
Chuan-Yue Yang, National Taiwan University, Taiwan

Special Thanks

Ludovic Chacun
Liliana Cucu-Grosjean
Christine Rochange
Jean-Luc Scharbarg

Table of Contents

Allocation-Site Aware Shape Analysis and Applications in Hard Real-Time Systems

Jörg Herter
Saarland University, Saarbrücken, Germany
jherter@cs.uni-saarland.de

Abstract

Shape analysis aims at determining invariants of heap-allocated structures that arise during the execution of a program. Current shape analysis techniques are stateless, i.e. they only model the structures arising on the heap and completely ignore their memory locations and where they were allocated. This paper proposes an extended, allocation-site aware shape analysis and briefly sketches fields of applications for such an analysis in the area of (hard) real-time systems.

1 Introduction

Shape analysis denotes a static program analysis that determines the *shape* of—or invariants that hold for—the heap at the different program points. Most recent approaches to shape analysis rely either on separation logic [8] to express inferred properties of structures arising on the heap [2], or they model the heap by 3-valued logical structures [9]. The commonality of all approaches is that they are *stateless*. I.e. they only model *what* heap structures may arise, not *where*, i.e. at what memory address, they reside on the heap nor *where* and *when*, i.e. at what allocation site and which invocation thereof, they were allocated.

For the current field of applications for shape analyses like checking data structure invariants [9, 2] and memory safety or even verifying partial program correctness [7], information about where and when heap objects were allocated is not required and it may hence be safely abstracted from in order to increase performance. However, in a real-time setting, applications for allocation-site aware shape analyses arise. Consider for example dynamic memory allocation in hard real-time applications. To enable tight bounds on the worst-case execution-times (WCET) of such programs, a WCET analysis must be able to correctly classify most accesses to heap objects as cache hits or cache misses. The *cache mapping* of objects depends on their addresses in memory, i.e. *where* they reside on the heap.

More concretely, we are currently aware of three approaches to enable the determination of tight WCET bounds for programs using dynamic memory allocation. Schoeberl proposes to use predictable hardware caches to separate dynamically and statically allocated objects [10]. To support this approach, a static analysis would have to associate (heap) objects with allocation sites (or just allocation technique: static or dynamic) to decide in which hardware cache an object may reside. Herter et al. propose to use a predictable memory allocator that takes as an additional argument the cache set to which the returned address shall be mapped [5]. As a result, the cache set mapping becomes explicit and statically known. However, for a WCET analysis to benefit from this, heap objects need to be associated with invocations of the dynamic allocator, i.e. *where* and *when* they were allocated. The third approach is only applicable to a subset of hard real-time applications with statically derivable regularities in allocation behavior and aims at removing dynamic allocation completely by replacing it by precomputed memory addresses [4, 3]. This approach heavily relies on a precise static analysis of the program to enable the computation of good memory addresses. It also requires that heap objects can be associated with allocation sites. While a data structure analysis [6] can be used to connect heap structures with allocation sites, an allocation-site aware shape analysis as proposed in this paper can yield more precise information resulting in a more efficient set of precomputed memory addresses.

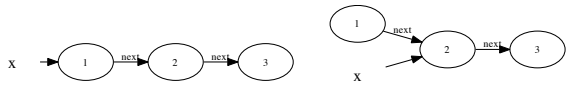
The remainder of this paper is organized as follows. Section 2 briefly introduces the framework for stateless shape analysis via 3-valued logic as proposed in [9]. In Section 3, we sketch how allocation-site awareness can be incorporated into this framework. Section 4 discusses static program analyses that would be enabled by or at least profit from an allocation-site aware shape analysis.

2 Shape Analysis using 3-valued Logic

This section briefly summarizes the framework for shape analysis via three-valued logic. For a detailed discussion, we refer to [9].

Two-valued logical structures can be used to describe concrete heap states. Each heap allocated object is represented by a logical individual, each pointer variable by a unary predicate that evaluates to true iff its argument

is the individual representing the heap object to which the variable points. Field pointers referencing one heap object from another are analogously modeled by binary predicates. Additional so-called *instrumentation predicates* can be defined to increase precision or performance of the analysis. Properties of the heap can be formulated as logical formulæ and checked by evaluating their defining formulæ on the logical structure describing the current heap state. Effects of program statements on the heap state are captured by predicate-update formulæ that state how predicates are updated to yield a structure describing the heap state after execution of a program statement. Figure 1(a) shows a graphical representation of a logical structure describing three objects organized in a singly linked list. Logical individuals are depicted as circles, predicates evaluating to *true* as arrows. Predicates evaluating to *false* are not drawn.



(a) (b)

Figure 1. Two shape graphs each depicting 3 objects organized in a singly linked list.

Applying the effects of the program statement $x = x{\rightarrow}next$; modeled by predicate-update formulæ

$$\begin{aligned} x(v) &\leftarrow \exists u.x(u) \wedge next(u,v) \\ next(u,v) &\leftarrow next(u,v) \end{aligned}$$

yields a structure as depicted in Figure 1(b).

A shape analysis of a given program/method can then be implemented as a fixed point computation collecting for each program state the set of logical structures describing all heap states that may arise there, starting with some initial heap description for the starting point of the program/method. However, an unbounded number of concrete heap description may arise at program points. We therefore introduce abstract heap descriptions using three-valued logical structures that can themselves represent a possibly infinite number of concrete two-valued logical structures. A concrete logical structure is abstracted by partitioning the individuals into equivalence classes such that all individuals within one class yield the same truth values for a predefined set $\mathcal{A}$ of *abstraction predicates*. The individuals of the abstracted structures correspond to these equivalence classes. Abstract individuals that may represent more than one concrete individual are called *summary nodes*. Predicates not in $\mathcal{A}$ need to be reevaluated and may evaluate to the indefinite truth value $1/2$ iff not all concrete individuals summarized by the abstract individual evaluate to the same definite truth value. Abstracting the structure from Figure 1(a) under $\mathcal{A} = \{x\}$ results in the 3-valued logical structure depicted in Figure 2, where dotted arrows represent predicates evaluating to $1/2$ and summary nodes are drawn doubly circled.

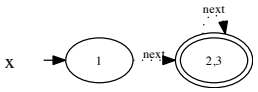


Figure 2. Abstract shape graph embedding in particular the structure from Figure 1(a).

To model the effects of program statements on abstract heap descriptions the same update formulæ as in the concrete setting are used and simply evaluated using 3-valued logic. However, to increase precision, before applying update formulæ the *relevant* parts of the structure are concretized (*focus* or partial concretization). As focusing may generate contradicting or less precise structures, after application of the update formula, the resulting structures are *coerced* into more precise structures and contradicting structures are completely removed.

3 An Allocation-Site Aware Shape Analysis

In order to make the previously described framework allocation-site aware, we associate with each heap object *where* and *when* it was allocated. The number of allocation sites is statically known and for most programs very small. Hence, to model *where* an object was allocated, we introduce additional unary predicates $alloc_m \in \mathcal{A}$ such that $alloc_m(u) = 1$ iff u was allocated at program location m. Furthermore, to model *when* the object was allocated, we construct a function $t^\sharp : \mathcal{U} \mapsto \mathbb{N}$ that maps individuals of a concrete structure to invocations of an allocation site. In an abstract structure, we map to intervals of possible invocations: $t : \mathcal{U} \mapsto \mathbb{I}$, where the set of intervals is defined as $\mathbb{I} = \{[l,u] \,|\, l \in \mathbb{N} \wedge u \in \mathbb{N} \wedge l \leq u\}$. Analogously, we can add functions $s^\sharp$ and s to associate heap objects with their (requested) sizes. Summarization of two individuals v_1 and v_2 is adapted as follows. Let the new summary node be v_{sm}, then $t(v_{sm}) = t(v_1) \sqcup t(v_2)$ and $s(v_{sm}) = s(v_1) \sqcup s(v_2)$ where $[l_1,u_1] \sqcup [l_2,u_2] = [\min\{l_1,l_2\}, \max\{u_1,u_2\}]$. The logical predicates are reevaluated as in the stateless framework.

Consider the C program given in Listing 1. Figure 3 shows an abstract allocation-site aware shape graph describing the possible heap states occuring after executing line 4. Being more precise than existing data structure analysis, we can identify two data structures and associate their objects precisely with the same occurrence of malloc in program line 12.

In a real-time setting, shape analysis can be performed arbitrarily precise. As in the general setting, we can add instrumentation predicates to increase precision, but we can also deactivate abstraction, i.e. summarization of individuals, completely as no unbounded structures may arise due to known loop and recursion bounds. However, abstract heap structures are still desirable as they may lead to significantly shorter analysis time. The following set of (instrumentation) predicates and additional

Listing 1. C program working on linked lists

```
1  int main() {
2      list * p = buildList(16, ...);
3      list * data = buildList(256, ...);
4      list * x = data;
5      ...
6  }
7  list * buildList(int size, ...) {
8      list * result;
9      ...
10     while(...) {
11         ...
12         ... = malloc(sizeof(list));
13         ...
14     }
15     return result;
16 }
17 struct dll_el * copy( struct sll_el * src ) {
18     struct dll_el * result;
19     ...
20     while ( src != NULL ) { /* loop bound exactly 256 */
21         ... = malloc(...);
22         ...
23         free(...);
24         ...
25     }
26     ...
27     return result;
28 }
```

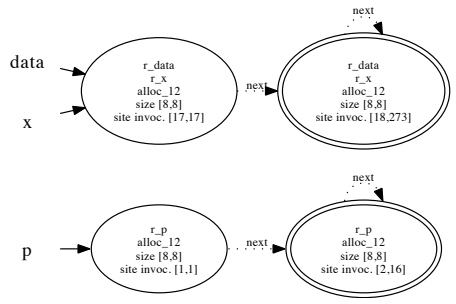


Figure 3. Analysis result after execution of line 4. `site invoc` **corresponds to our** t **function.**

precision increasing techniques have shown good trade-offs between precision and complexity of allocation-site aware shape analyses. To separate different data structures, a predicate $r_x(v)$ modeling reachability from program variables is used: $r_x(v) := \exists u.x(u) \wedge fr(u,v)^*$, where x and fr are predicates corresponding to pointer variables and field references, respectively. Deallocated objects are not removed but marked as freed by a unary predicate $deallocated(v)$. We further increase precision of partial concretization w.r.t. numeric intervals by allowing the analysis to mark predicates modeling field references with superscripts $^<$ and $^>$, indicating that, iff the predicate evaluates to true, both arguments to the predicate are allocated directly after or before each other at the same allocation site. We also introduce an additional abstraction technique that substitutes in intervals of length 1, as in $[5,5]$, the numerical value by newly introduced variables, yielding in the example the interval $[i,i]$. This enables embedding of structures that differ only in numerical values used as interval bounds. The shape graph depicted in Figure 4 is one of the 3 shape graphs arising after execution of line 25, when embedding is extended as described.

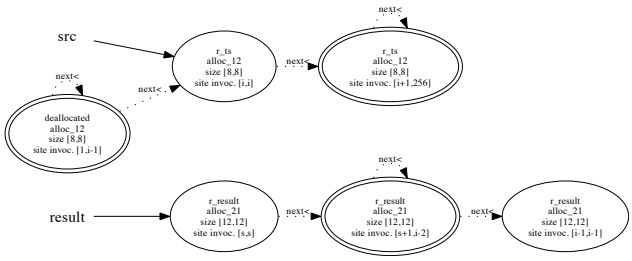


Figure 4. A possible heap state at line 25.

4 Applications

Applying an analysis as discussed in the previous section yields sets of allocation-site aware shape graphs for an analyzed program. Information extracted from these graphs can be used to enable program transformations that increase timing predictability and can even constitute analysis results for other, subsumed analyses. This section gives an overview on how information from shape graphs can be of benefit and sketches the applications for allocation-site aware shape analysis we identified so far.

4.1 Allocation Behavior Analysis

Our main motivation was to enable a more precise timing analysis for hard real-time applications. For programs with statically derivable regularities in object lifetimes, replacing dynamic memory allocation by a precomputed static allocation scheme yields many advantages. The memory addresses of heap allocated objects become known to the timing analysis and unpredictability introduced by the memory allocator is removed together with the allocator itself.

The precomputation of suitable memory addresses for heap objects as proposed in [3] relies on a formal description of a program's allocation behavior. This formal description is given by a six-tuple, $(M,U,L,A,\mathcal{C},\mathcal{B})$. M is the set containing all allocation sites and U contains upper bounds on how often each allocation site may be reached, i.e. how often this function call may be invoked. Additional knowledge about the relations between elements of U, such as $u_1 < u_2$, is collected in the set L. For each allocation site m, a function f_m is constructed such that $f_m(i)$ evaluates to an interval describing the size of the memory block requested the i-th time allocation site m is reached. A is the set of all such functions. The set R where $R = \left\{(m,i) \mid m \in M \wedge i \in \mathbb{N}^{\leq u_m \in U}\right\}$ contains all allocation requests that may occur during program execution. C is a conflict function $\mathcal{C} : 2^R \mapsto \{0,1\}$ that

evaluates to 1 iff its argument requests at least two memory blocks with overlapping lifetimes. To exploit simple cache placement heuristics, a bias function $\mathcal{B}$ is given as $\mathcal{B} : (R \times R) \mapsto \{0, 1\}$ where $\mathcal{B}(r_1, r_2)$ evaluates to 1 iff the block requested in r_1 is likely to be accessed prior to the one requested in r_2. While M can be directly extracted from the program code, U and L are often provided by the user. A is constructed from M, L, and the requested sizes. These sizes, the conflict function $\mathcal{C}$, and the bias function $\mathcal{B}$ have to be derived by a static program analysis. The size of a heap object—or requested memory block—is explicitly stored in the shape graphs. Functions $\mathcal{B}$ and $\mathcal{C}$ can be extracted from an allocation-site aware shape analysis as follows. $\mathcal{B}(r_1, r_2)$ evaluates to 1 iff there exists a field-pointer predicate evaluating to true for the individuals representing r_1 to r_2, $\mathcal{C}$ evaluates to 1 iff representatives of at least two elements of its argument set are present in the same shape graph and none is marked as deallocated.

4.2 Cache Analysis for Heap Allocated Objects

For programs from which we cannot statically remove dynamic memory allocations, a cache-aware predictable memory allocator may be used. Such an allocator as proposed in [5] can be guided with respect to the cache set mapping of returned addresses via an additional cache set argument. Knowing to what cache sets the memory locations of heap objects are mapped, cache and subsequent WCET analyses may be able to predict cache hits or cache misses for accesses to dynamically allocated objects. Failing to be able to correctly classify a significant number of memory accesses as cache hits or misses would result in high overestimations of a program's WCET. A shape analysis as proposed in this paper can be used to automatically find suitable cache set arguments for allocated objects by extracting program logical structures from the resulting shape graphs (see Section 4.3) and applying a suitable cache set mapping strategy to the respective structures. Also, combining a cache analysis [1] with an allocation-site aware shape analysis is current on-going work.

4.3 Combined Data Structure and Escape Analysis

Data Structure Analysis attempts to identify disjoint instances of program logical data structures and their internal and external connectivity properties [6]. An escape analysis categorizes objects into escaping and not escaping their allocating function [11]. An object is said to escape the function it was allocated in if it may still be accessible after returning from this function. The aim of such an analysis is typically to identify objects that can be allocated on the stack instead of the heap to increase program performance.

Both analysis results can be extracted from allocation-site aware shape graphs. A program logical data structure can always be defined as a connected component of the shape graph. With more knowledge about the data structures used in a program, we can even introduce new predicates to more precisely associate heap objects with program logical data structures. To identify escaping objects, we check within the shape graphs occurring at the exit point of functions whether objects allocated within the function are reachable from returned, static or class objects or also arguments to the analyzed function. Objects passed as arguments to functions called within the analyzed method also escape. Reconsider Figure 3 describing all possible heap states after execution of line 4 of our example code, where an allocation-site aware shape analysis was able to separate all heap allocated objects into two disjoint data structures. Existing data structure analyses like [6] yield a much less precise overapproximation associating all heap objects with one structure.

5 Conclusions

The current stateless shape analysis framework via 3-valued logic can be extended to track information about where and when heap objects were allocated as well as their respective sizes. For applications of shape analysis considered within the community so far, this additional information is not necessary and only tends to increase analysis times. However, for hard real-time programs, applications have emerged that depend on this additional information. Additionally, performance of static analyses is less critical in this setting as the analyzed programs are normally less complex and higher analysis times are justifiable.

References

[1] M. Alt, C. Ferdinand, F. Martin, and R. Wilhelm. Cache behavior prediction by abstract interpretation. In *SAS '96*, London, UK, 1996. Springer-Verlag.

[2] B.-Y. E. Chang and X. Rival. Relational inductive shape analysis. In *POPL '08*, New York, NY, USA, 2008. ACM.

[3] J. Herter and S. Altmeyer. Precomputing memory locations for parametric allocations. In *WCET'10*, 2010.

[4] J. Herter and J. Reineke. Making dynamic memory allocation static to support WCET analyses. In *WCET'09*, 2009.

[5] J. Herter, J. Reineke, and R. Wilhelm. CAMA: Cache-aware memory allocation for WCET analysis. In *Proceedings Work-In-Progress Session of the 20th Euromicro Conference on Real-Time Systems*, 2008.

[6] C. Lattner and V. Adve. Data structure analysis: A fast and scalable context-sensitive heap analysis. Technical report, University of Illinois at Urbana-Champaign, 2003.

[7] T. Lev-Ami, T. Reps, M. Sagiv, and R. Wilhelm. Putting static analysis to work for verification: A case study. In *ISSTA*, 2000.

[8] J. Reynolds. Separation logic: A logic for shared mutable data structures. IEEE Computer Society, 2002.

[9] M. Sagiv, T. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. *ACM Transactions on Programming Languages and Systems*, 24(3), 2002.

[10] M. Schoeberl. Time-predictable cache organization. In *STFSSD '09*, Washington, DC, USA, 2009.

[11] J. Whaley and M. Rinard. Compositional pointer and escape analysis for java programs. *SIGPLAN Not.*, 34(10), 1999.

Model-based approach for IMA platform early exploration

Michaël Lafaye[a,b], David Faura[a], Marc Gatti[a], Laurent Pautet[b]

[b] TELECOM ParisTech
LTCI
46 rue Barrault
75634 Paris Cedex 13, France
{lafaye, pautet}@telecom-paristech.fr

[a] THALES Avionics
ACS/DTEA
18 avenue du Maréchal Juin
92366 Meudon-la-Forêt Cedex, France
{david.faura, marc-j.gatti}@fr.thalesgroup.com

Abstract

The avionics platform are now developed according to the Integrated Modular Avionics concept, allowing one processing module to host some applications in order to reduce weight, space and costs. This concept increases the development and certification complexity, while time to market tends to decrease. So new development processes are needed. Model-based approaches are now mature enough to design embedded critical systems and perform architecture exploration.

In this paper we propose a new modeling approach that aim to size an execution platform architecture (OS and hardware) according to the applications it has to process, and achieve early platform exploration.

Keywords : modeling, avionics systems, real-time, simulation, AADL, SystemC

1. Introduction

Avionics systems are critical real time systems composed of applications, real-time operating system and hardware modules. To reduce the environmental footprint in term of weight and space, and also the costs, the Integrated Modular Avionics (IMA) concept was developed in the 2000s. It defines integrated architectures, where one calculator hosts some applications. So the number of modules aboard is reduced. Following this evolution, suppliers developed network architectures in which modules are interconnected and communicate through a deterministic network. However, this evolution entails an increase of complexity in avionics platform design, verification and certification processes, while time to market tends to decrease. These developments require an early modeling of the system to validate and maximize the use of the future platform, while ensuring the critical level required by current standards in aviation (DO-178B, MILS-CC…).

To model IMA platform and perform early validation, **M**odel-**D**riven **E**ngineering approaches [1,2] are now suitable to describe system high-level functionalities. Many projects aim at modeling these platform with **A**rchitectural **D**escription **L**anguages as AADL [3] or MARTE [4], or with synchronous languages such Lustre or Signal [5]. However, they often focus on the applications description, model the hardware as connected blackbox components with a few properties, and perform static simulation. Moreover, there actually is no automated process for complete platform modeling and simulation.

We propose a new modelling approach that aims to design a complete avionics system (hardware, operating system and the applications), and perform dynamic simulation. It is a component-based approach, relying on two languages and taking advantages of both: AADL and systemC. AADL[8] is, as MARTE, an ADL particularly adapted for software architecture description [4,6], enabling the modeling of ARINC 653 embedded real-time system [7]. In view of the experience of partners who especially developed the ARINC 653 AADL annex and the Ocarina tool suite, we choose the AADL rather than MARTE. SystemC [9] is an IEEE standard widely used in industry for hardware platform description, and including a simulation kernel for architecture simulation and exploration.

In this paper we first present real-time system modeling works with ADL before introducing IMA concept, then we detail our method, and conclude with perspectives of our work.

2. State Of The Art

Some projects aimed at modeling real time critical systems with ADL. The SPICES project [3] and MARTES project [4] both tried to use a combination of an ADL (MARTE for MARTES, the AADL for Spices) and the systemC language. Their goal was to model at high level a real-time system, and generate by code translation a systemC platform. The idea of using

such a combination seems relevant, but the part of code translation from the ADL to systemC platform model is a harder operation, because there are lots of cases to manage. Moreover, the systemC hardware description does not allow enough accuracy in execution platform performances analysis for our approach.

Our approach is also partly based on the ARINC653 system modeling works with AADL, that led for the development of the AADL ARINC653 annex [7]. It enables the description of the spatial and temporal partitioning method, secure inter and intra partitions communications, and applications deployment on the execution platform. However, there is no behavior specification for the hardware part, and no dynamic simulation. Our approach looks at fulfill these lacks.

3. Integrated Modular Avionics Platform

An IMA platform, as illustrated in figure 1, is composed of avionics applications running on execution modules including embedded operating system and the underlying hardware. These modules communicate through a deterministic network, the AFDX (Avionics Full DupleX).

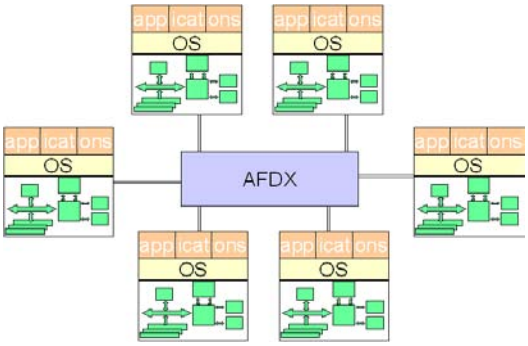


Figure 1. IMA platform

A processing module hosts some applications at different criticality levels, it is then necessary to respect safety constraints. That's why OS ARINC 653 standard was defined, which specifies space and time partitioning. To ensure space partitioning and prevent failure propagation, each application is enclosed in one or some partitions, isolating it from each other. Each partition is bound to a part of memory, so it only access its own memory area (figure 2). The standard also defined secure intra and inter-partitions communications. To ensure time partitioning, each partition has its own execution time window, during which the application has access to all dedicated resources (processing, memory, dedicated I/O, etc.).

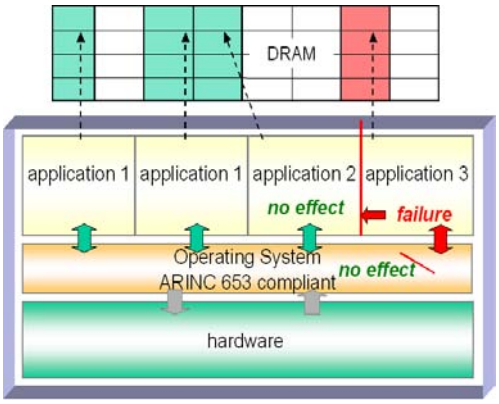


Figure 2. ARINC 653 spatial partitioning

4. IMA Platform Modeling

4.1. Overview

Model-Driven Engineering approaches are now mature enough to serve as a basis for building embedded systems and perform early validation. They are especially suitable for modeling high-level functional architecture (description of the functionalities offered by the system) and logical architecture (description of how the system is structured into logical components cooperating by communications) [10]. But at platform architecture level, these approaches describe both hardware and software as static blackbox elements with some properties.

Some projects [3,4,5] aim at building more accurate platform models, but they mainly focus on the software behavior, and model hardware as one or a few blackbox components without behavior information. They after perform static analysis on the model. So there is no method to retrieve dynamic performances from the hardware in order to validate it according to the applications requirements.

Our method develops a new approach for avionics platform modelling, that refines architecture description at platform decomposition level, specially the hardware. In order to refine this later, our approach models and sizes the execution platform according to applications requirements. As we can see in figure 3, this approach consists of different tasks:

- system high level modeling;
- applications characteristics extraction;
- platform generation according to requirements;
- platform simulation and performances analyses.

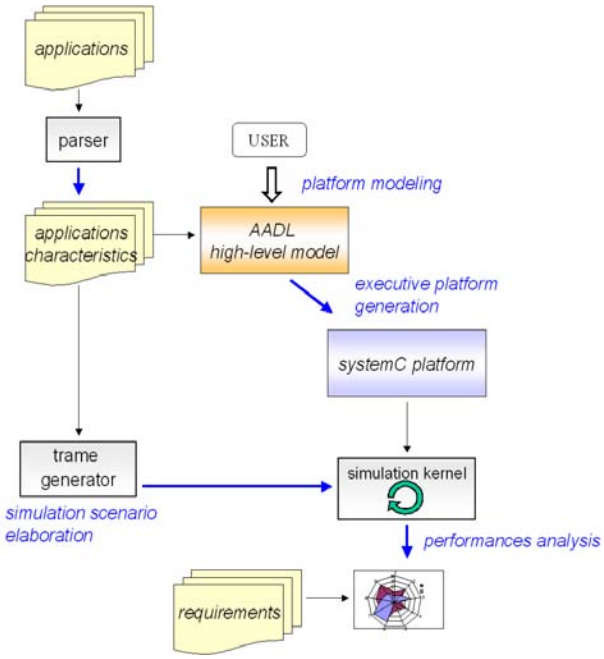


Figure 3. Modeling Approach

4.2. Applications modeling

An important point at this step is that we have not necessarily access to the applications code, so we can not directly generate the applications characteristics files (Figure 4). Consequently, we define the main applications characteristics we need in order to generate stimuli to simulate the platform. These characteristics are for example parallel code percentage, I/O access, processes duration and deadline…which can be obtained by code profiling or extracted from the applications configuration files.

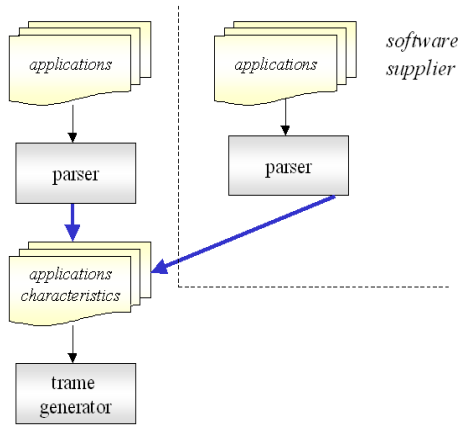


Figure 4. Application modeling.

These characteristics are first used to model the applicative part of the platform with the AADL. Partitions are modeled with the AADL processes, and ordered according to a given pattern called Major Frame, as in figure 5, where P1 represents the partition 1 and so on. The AADL threads represent the ARINC 653 processes of the applications. They are configured (deadline, priority etc.), bound to an AADL processor and a part of the main memory. Previous works defined rules to model ARINC653 system with the AADL [7].

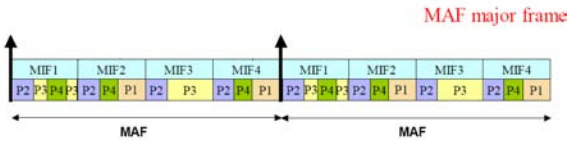


Figure 5. Example of Major Frame

These applications characteristics files is our input. We defined a set of macro-instruction (for example dram_read_command) which target the different modules of the platform. Thus we translate the different characteristics into these systemC macro-instruction. These instructions will be injected into the simulation kernel when simulating the future platform.

4.3. High level system modeling with AADL

Contrary to the applications, described with AADL threads, the operating system is defined by some properties dispatched in the different hardware components. For example, scheduling policy is set in the processor module, partition security level is defined in the virtual processor, etc. To model an ARINC 653 operating system, we use the AADL 653 annex, and the method described in this article [7].

Each hardware component is modeled with the AADL corresponding component, or with the device element. Some more complex components, like network, are modeled as a sub-system containing other components. We first model hardware component as a pseudo blackbox element, where behavior is not defined. We set interface information like ports, and a few properties (memory size, bus transfer latency etc.). In order to refine those hardware properties, we defined or completed some AADL property sets, by introducing behavior and specific properties like cache hit rate for cache component, or refresh time for DRAM component.

The user models the avionics system with the AADL, and choose which viewpoint(s) will be set when analyzing the platform. Viewpoints can be for example timing performance, power consumption etc., and enable the platform investigation under different angles. We then parse the AADL model to retrieve properties, connections and deployment information, that are configuration information for the next step.

4.4. Platform integration by generation

We developed a database of configurable systemC components. They are automata which can be configured with properties and viewpoints. For example, on the figure 5, the automata of the DRAM was configured according to the timing viewpoint, and then with timing properties (tRefresh = refresh latency etc.).

In order to generate the refined executive platform, we retrieve from the AADL model the different hardware properties and connections, configure the corresponding systemC element, and connect them.

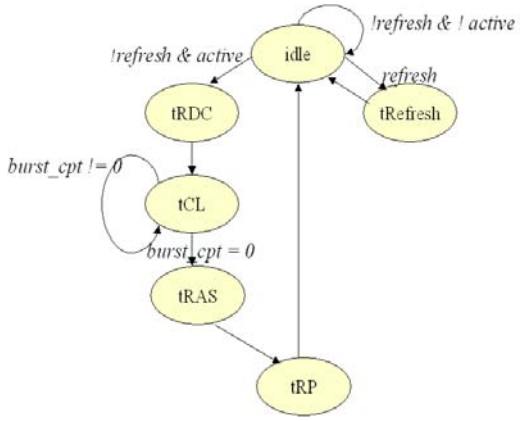


Figure 6. Dram automata example with timing annotations

5. Platform Simulation and Results

Currently, this is a work in progress. However, we have already encouraging results. The AADL part of the process has been specified and is under development, while systemC main hardware have been developed (behaviour, main properties and communication interface), as the first blocks of a future model of an ARINC653 OS (scheduler, timer, switch partition and context). In order to test and refine these elements, we also developed a minimum hardware platform and a basic scenario that simulates it in order to test and refine the elements behaviour and the platform communications.

To test our future platform, we defined a simulation and performances analysis method: to see if the hardware platform built is compliant with the application requirements, we will perform a simulation using systemC kernel, which will take the generated platform, the viewpoint(s) set, and will apply the simulation(s) scenario(s). The user will be able to analyze the platform performances by examining performances graphs and simulation traces. If the execution platform is not compliant with the applications requirements, we then will investigate and try to refine or modify one or some components.

6. Conclusion

Current early platform validation methods center on software modeling, regarding the hardware as blackbox components which can't be dynamically simulated. We have proposed a new early validation approach, that aims to model a complete avionics platform, software and hardware. Our method should automatically generates hardware and simulation scenario. It should also enables a dynamic simulation of the platform, and analyzes its performances according different viewpoints (timing, power consumption or safety). It takes advantages from the AADL, particularly adapted for software architecture modeling, and from systemC, industrial standard for hardware architecture description.

To validate the accuracy of our modelling methodology, we will first model electronic evaluation boards. We will afterwards model a complete IMA platform to compare the model performances with the experimental results. Otherwise, we will connect with existing model-driven engineering methods and improve the platform development process.

References

[1] J.A. Estefan. "Survey of model-based systems engineering (MBSE) methodologies". Technical report, *INCOSE MBSE Focus Group*, may 2007

[2] Bernhard Schätz, Manfred Broy, Franz Huber, Jan Philipps, Wolfgang Prenninger, Alexander Pretschner, Bernhard Rumpe, "Model-Based Software and Systems Development – a white paper", 2004

[3] Support for Predictable Integration of mission Critical Embedded Systems project (SPICES), 2009 http://www.spices-itea.org

[4] Model-Based Approach for Real-Time Embedded Systems development project (MARTES), 2007. http://www.martes-itea.org/

[5] C. Brunette, R. Delamare, A. Gamatié, T. Gautier, J-P. Talpin, "A Modeling Paradigm for Integrated Modular Avionics Design", IRISA report, 2005.

[6] P. Dissaux, F. Signhoff, "the AADL as a Pivot Language for Analyzing Performances of Real Time Architectures", *4th European Congress ERTS Embedded Real Time Software*, 2008.

[7] J. Delange, L. Pautet, A. Plantec, M. Kerboeuf, F. Singhoff, F. Kordon, "Validate, Simulate and Implement ARINC653 systems using the AADL", *CM SIGAda Ada Letters*, 2009.

[8] AADL Portal at Telecom Paristech : http://aadl.telecom-paristech.fr/

[9] Open SystemC Initiative. IEEE 1666: SystemC Language Reference Manual, 2005. www.systemc.org.

[10] L. Cai and D. Gajski, "Transaction Level Modeling: An Overview", *Center for Embedded Computer Systems, University of California*, 2003

Improved sampling for statistical timing analysis of real-time systems *

Dorin Maxim, Luca Santinelli and Liliana Cucu-Grosjean
INRIA Nancy-Grand Est, f rstname.lastname@inria.fr

Abstract

The guaranteeing of timing constraints is the main purpose of analyses for real-time systems. The satisfaction of these constraints could be verified deterministically using worst-case scenarios that introduce a certain pessimism. This pessimism could be decreased by using statistical estimations of certain parameters as it is the case for worst-case execution times. In this paper, we address the problem of sampling applied to the distributions of worst-case execution times. The presented sampling techniques are investigated in order to provide the respect of timing constraints and to insure a low level of pessimism.

1 Introduction

Real-time systems are, generally, embedded and they are interacting with the environment. The performances of such interactions are then analyzed, not only from the point of view of their correctness, but also from the perspective of time. The timing analysis of such systems has been extensively studied by considering worst-case values that induce a certain pessimism. Unfortunately not all real-time systems could afford this pessimism and for these cases other approaches should be used (statistical and probabilistic approaches, agent learning, game theory, etc). In this paper we deal with statistical approaches for real-time systems with variable worst-case execution times. More precisely, we are interested in the obtention of the distributions for the worst-case execution times (WCETs) of tasks. We propose a sampling method that improves the results presented in [1] by decreasing the pessimism as well as the complexity of the associated timing analysis.

2 Related work

Statistical timing analyses provide the distributions of WCETs from large mass of data. These distributions could be empirical, thus easy to obtain but diff cult to use because of the large number of possible values for WCET. The number of possible values for the WCET has a direct impact on the complexity of the timing analysis that provides the probability of missing the deadline [2].

The distributions of WCETs could also be obtained using different statistical techniques like extreme values [3–5]. These techniques could introduce errors, thus different levels of conf dence and pessimism. We use here the concept of pessimism with the meaning def ned in [6].

Sampling techniques could be used on the distributions that have been already obtained using methods cited before. These techniques must be applied such that the real-time constraints are met and no relevant information is lost. For instance in [1] the authors propose a sampling technique [7] that decreases the number of possible values for the WCETs.

Contribution of the paper. We propose a re-sampling method that, although similar to the one in [1], represents an improvement since it decreases signif cantly the pessimism by minimizing the probabilities associated to the largest values of WCETs.

Organization of the paper. The paper is organized as follows. In Section 3 we describe the problem to solve and the associated notations. Our improvement on the real-time sampling is presented in Section 4 and an example of its utilization is provided in Section 5. We conclude in Section 6.

3 Problem statement and associated model

We deal with the preemptive f xed-priority scheduling of asynchronous periodic tasks with variable execution times on one processor. We consider $\tau = \{\tau_1, \tau_2, \cdots, \tau_n\}$ a set of n periodic tasks ordered according to their priority.

Each task is characterized by an offset O_i, an exact inter-arrival time T_i, a relative deadline D_i and a probability of meeting the deadline p_i. It means that the j^{th} job of τ_i is released at time instant $O_i + (j-1)T_i$ and it must f nish its execution by time instant $O_i + (j-1)T_i + D_i$. Among all (representative) jobs of a task at least p_i of them must f nish their execution by their deadline. When for a given priority assignment a job will always miss its deadline (in

*This work was partially founded by the European Community's Seventh Framework Programme FP7/2009-2013 under grant agreement no. 249100.

all possible scenarios) then the job is discarded. We denote by $\mathcal{C}_i$[1] the random variable providing the possible values for worst-case execution time of task τ_i, see Equation (1). It is assumed that the random variables giving the worst-case execution times are independent.

$$\mathcal{C}_i = \begin{pmatrix} c_k \\ P(C = c_k) \end{pmatrix}_{k \in \{1, \cdots, k_i\}} \quad (1)$$

In Equation (1) $c_k \in [c_i^{min}, c_i^{max}]$ and $k_i \in \mathbb{N}^*$ is the number of values that the random variable $\mathcal{C}_i$ has. We consider that c_i^{min}, c_i^{max} are given, thus known.

We denote a task τ_i by $(O_i, \mathcal{C}_i, T_i, D_i, p_i)$. For such a task and under a preemptive f xed-priority scheduling policy, Diaz et al. [2] provides a probabilistic timing analysis for the response time of any of its jobs. This analysis takes as input the distributions $\mathcal{C}_i$ of all tasks and its complexity is directly related to the number of possible values of $\mathcal{C}_i$. This relation makes such analysis impractical except for simple scenarios. For instance, for two discrete random variables $\mathcal{C}_i$ and $\mathcal{C}_j$, during each iteration of the timing analysis we obtain, for any convolution, a new random variable that may have, in the worst-case, $k_i * k_j$ values.

A solution to this problem is to reduce the number k_i of possible values of $\mathcal{C}_i$ as long as the real-time constraints are meet. In the rest of this paper we propose such solution by means of sampling. Thus, the inputs of our problem are distributions of worst-case execution times and their obtention is beyond the purpose of this paper.

We investigate here sampling problems in real-time scenarios, in particular the re-sampling of distributions of worst-case execution time and we give an answer to the question: *Given a distribution of n values, how do we select k significant values out of n and how do we redistribute the probabilities so that the analysis is still accurate?* Such a new distribution has to offer the same real-time guarantees while reducing the available information by re-sampling.

In probabilistic analysis, the response times are random variables while real-time constraints are expressed in terms of probabilities of deadline misses. The response time $\mathcal{R}$ is one of the metrics applied to evaluate the probabilistic real-time analysis. Based on that, Diaz et al. in [2] have def ned relationships in order to tackle with the pessimism in probabilistic analyses. In particular we are interested in the relationship "greater than or equal to" which is def ned among random variables.

Definition 3.1 (Exact probabilistic analysis)**.** *An exact random variable defines the exact results of a probabilistic analysis.*

A random response time $\mathcal{R}'$ is pessimistic when it is greater than or equal to the exact one $\mathcal{R}$. The challenge with the re-sampling is how to perform approximations with probabilistic analysis parameters, while guaranteeing that the resultant response time are pessimistic or identically distributed, thus making the approximated analysis safe.

Any modif cation of an exact random variable has to be related to the exact random variable and its results in order to verify if the analysis remains safe and the level of pessimism of the resulting analysis.

[1] In this paper we utilise calligraphic letters to denote random variables

4 Real-time sampling

Refaat et al. propose a sampling method that simplif es the distribution $\mathcal{C}_i$ of a task τ_i [1]. They select k samples from the k_i available in the original distribution. The resulting distribution keeps the largest value of the original distribution as well. The k values are kept with their respective probabilities, while the remaining probabilities (the ones associated to the values which have not been selected) are accumulated to the largest value of the distribution. See Figure 1(b) for exemplif cation.

We propose a new sampling method that improves the method proposed in [1]. Our method is based on a more accurate sample selection and a better probability redistribution among the chosen samples. First the k required values are sampled, then the probability of the old distribution is re-distributed among those new samples. In this work we assume that k is given and its obtention is beyond the purpose of this paper. By better probability re-distribution we intend a reduced pessimism as we will clarify later on.

The re-sampling mechanism we are proposing is named High value Re-Sampling (HiRS).

4.1 Selecting the k samples

We present here the selection of the k samples out of n for a given distribution $\mathcal{C}_i$. Our proposal consists of *choosing the k values with the highest probabilities*. If the largest value is among them, then the selection mechanism ends. Otherwise, we keep the $k-1$ values with the highest probabilities while the k-th sample is the one with the largest execution time. Thus, it is guaranteed that the largest execution time value is in the new distribution and a real-time analysis can be carried out.

By selecting the values with the highest probability, we provide a new distribution with the most representative samples. This new distribution has a reduced amount of probability which has to be re-distributed in the new random variable (reduced with respect to the one obtained in [1]).

4.2 Probability re-distribution

The main contribution of our re-sampling mechanism resides in *distributing the remaining probabilities to the k se-*

lected values instead of accumulating and adding them to the probability of the highest value. Thus, we increase the accuracy of the analysis and in the same time we decrease the pessimism. Moreover, the mechanism of distributing the probabilities to the k selected values is thus improved with respect to [1].

We detail now the re-sampling mechanism. The probability distribution mechanism that we propose is described by Algorithm 1.

Algorithm 1 Algorithm for re-distributing the probabilities to the k selected values

Input: $\mathcal{C}_i$ and k selected values;
Output: $\mathcal{C}_i^{new}$;
  $j = 1$;
  $m = 1$;
  $p = 0$;
  while $j \leq k_i$ **do**
    $p = p + P(\mathcal{C}_i = c_j)$;
    if j is a selected value **then**
      $c_m^{new} = c_j$, $P(\mathcal{C}_i^{new} = c_j) = p$ and $p = 0$;
      $m = m + 1$;
    end if
    $j = j + 1$;
  end while

Intuitively, Algorithm 1 is cumulating to the probabilities $P(\mathcal{C}_i = c_i)$ associated the values of $\mathcal{C}_i$ starting from c_{i_1-1} to c_{i_2}, where i_1 and i_2 are two consecutive selected values. The obtained cumulated probability is associated to the value c_{i_2} in the new random variable $\mathcal{C}_i^{new}$.

We continue this process until we pass trough the entire random variable. At the end of this process we will have a resulting random variable with k values.

By the combination of the selection mechanism and probability distribution we only move towards the worst value a small part out of the total mass of the distribution. The resulting re-sampled distribution will have the worst value (the largest execution time) with a smaller probability than in [1]. So we are able to reduce the pessimism of the re-sampled distribution and include the samples with the highest probability. The re-sampling mechanism we have developed is illustrated in Figure 1(c).

4.3 Safety

The HiRS re-sampling mechanisms proposed has important results. The initial random variable $\mathcal{C}$ is the original distribution so it provides the exact results through the probabilistic analysis.

Lemma 4.1 (Safe re-sampling)**.** *The HiRS re-sampling mechanism is safe*.

Proof. The selection of the values for the re-sampled distribution does not affect the safety because the samples are a subset of the original ones. Instead, it is the probability re-distribution among the new samples to have an effect on the safety of the new distribution. In HiRS we cumulate the probabilities up to the largest values, so rightward with increasing ordered values. This results in a distribution in $\mathcal{X}'$ "greater than or equal to" the exact one $\mathcal{X}$. Since the response time $\mathcal{R}(\mathcal{X})$ (the response time $\mathcal{R}$ as a function the distribution $\mathcal{X}$) is a monotonic increasing function of $\mathcal{X}$, the new distribution results pessimistic with respect to the exact one, thus the analysis with the new distribution remains safe. Refer to Proposition and Theorem 1 in [2] for more details. □

The re-sampling, which changes the original probability distribution, cannot provide optimistic results because that would mean to neglect some information and than loose the safety of the analysis. The largest value of the original distribution has to be included in the f nal distribution being critical for the safety of the analysis. Indeed, removing that value would provide optimistic results with respect to the exact analysis.

From Lemma 4.1 and the results in [1] we can conclude that any probability re-distribution from left-to-right is safe because the unselected probabilities are moved to lager execution time values. On the other hand, the re-distribution right-to-left would produce optimistic results in terms of response time. Moreover, the results of the analysis with the re-sampled distribution obtained with HiRS represent an improvement of those obtained in [1]. Indeed, naming the distribution obtained with HiRS $\mathcal{X}'$ and the one obtained in [1] as $\mathcal{X}^{\mathcal{R}}$, it is that $\mathcal{X}^{\mathcal{R}} \succeq \mathcal{X}' \succeq {}^{2}\mathcal{X}$.

5 Example

To better understand the proposed sampling method we present an example. We consider a task with execution time given by a random variable with 10 values. Without loss of generality we consider these ten values as the integer numbers from 1 to 10. These values can be any numbers, we are only concerned to have them ordered in an increasing order, starting with the smallest value and ending with the largest value. The probabilities of these values are as follows:

$$\mathcal{C} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0.05 & 0.04 & 0.2 & 0.05 & 0.22 & 0.05 & 0.3 & 0.04 & 0.04 & 0.01 \end{pmatrix},$$

as it can be seen in Figure 1(a). The objective is to reduce this random variable to only $k = 4$ values instead of 10 in order to ease the next real-time analysis. According to

[2] $\succeq$ is the "greater than or equal to" relationship def ned by Diaz et al. in [2]

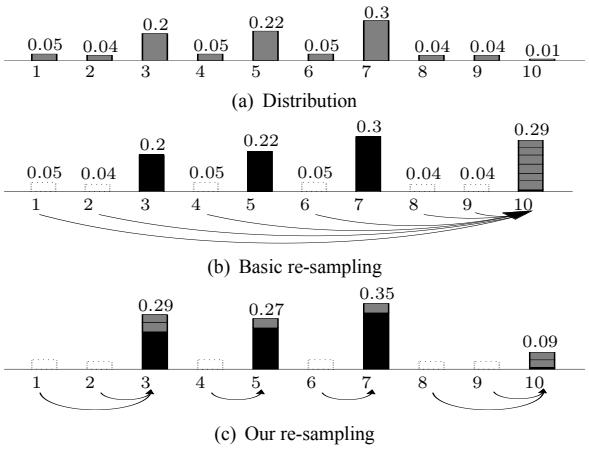


Figure 1. The example distribution with two different re-samplings.

our method, these 4 resulting values will be the last value of the original distribution (10-th value) and the 3 values with the highest probabilities, i.e., the 7-th value which has a probability of 0.3, the 5-th value which has a probability of 0.22 and the 3-rd value which has a probability of 0.2.

By applying the re-sampling method proposed in [1] we obtain a random variable (of 4 values) that has the following distribution:

$$\mathcal{C} = \begin{pmatrix} 3 & 5 & 7 & 10 \\ 0.2 & 0.22 & 0.30 & 0.29 \end{pmatrix},$$

see Figure 1(b). The samples according to the method presented in [1] should result from a random selection. In order to compare the results with our method, we keep the same values that we would have chosen with our solution. It is notable that the largest value of the distribution gets all the mass-probabilities of the values that were not selected to be kept in the new random variable. As a result, the worst value has an important associated probability, i.e., 0.29 in this case.

By applying the mechanism presented in Section 4, we are able to safely distribute the probability of the unused samples to the closest sample on the new distribution. For instance here the probabilities associated to the values 1 and 2 go to value 3 which results in a probability of $0.29 = 0.2 + 0.05 + 0.04$ in the new random variable. After the re-sampling, our task has now a worst case execution time given by the following random variable:

$$\mathcal{C}^{new} = \begin{pmatrix} 3 & 5 & 7 & 10 \\ 0.29 & 0.27 & 0.35 & 0.09 \end{pmatrix},$$

see Figure 1(c). The last sample results with a probability of 0.09, much lower than the one obtained by Refaat et al. [1]. Since that value represents the largest worst case execution time, then assigning a lower probability to this value makes our analysis less pessimistic in terms of response time and the resulting deadline miss ratio.

6 Conclusions and future work

In this work we have addressed the problem of re-sampling complex distributions of worst case execution times. The obtained distribution is less complex then the initial distribution, but it still has a high degree of accuracy and a low degree of pessimism, thus making it much more eff cient for the real-time analysis.

This work is our f rst step in studying the re-sampling effect on the response time of real-time systems. We believe that it is critical to keep the worst execution time value with the associated probability as small as possible. This aspect will be deeply investigated in future works. In the future we would also like to release some of the assumptions and derive the optimum value of k together with the optimal re-sampling whenever an execution time distribution is provided.

References

[1] Refaat, K.S., Hladik, P.E.: Eff cient stochastic analysis of real-time systems via random sampling. In: Euromicro Conference on Real-Time Systems, IEEE Computer Society (2010) 175–183

[2] López, J.M., Díaz, J.L., Entrialgo, J., García, D.: Stochastic analysis of real-time systems under preemptive priority-driven scheduling. Real-Time Syst. **40**(2) (2008) 180–207

[3] Edgar, S., Burns, A.: Statistical analysis of WCET for scheduling. In: 22nd IEEE International Real-Time Systems Symposium. (2001) 215–224

[4] Hansen, J., Hissam, S., Moreno, G.: Statistical-based WCET estimation and validation. In: 9th International Workshop on Worst-Case Execution Time (WCET) Analysis. (2009)

[5] Beirlant, J., Teugels, J., Goegebeur, Y., Segers, J., Waal, D.D., Ferro, C.: Statistics Of Extremes: Theory And Applications. Wiley (2004)

[6] Diaz, J., Lopez, J., M., G., Campos, A., Kim, K., Lo Bello, L.: Pessimism in the stochastic analysis of real-time systems: Concept and applications. In: 25th IEEE International Real-Time Systems Symposium. (2004) 197–207

[7] Fuller, W.A.: Sampling Statistics. Wiley (2009)

Maximum Access Delay Evaluation in an IEEE 802.11e/AFDX Hybrid Network

Bafing Sambou, Fabrice Peyrard, Christian Fraboul
IRIT/INPT-ENSEEIHT, University of Toulouse, France
2 rue Charles Camichel 31071 Toulouse, France
{Bafing-Cyprien.Sambou, Fabrice.Peyrard, Christian.Fraboul}@irit.fr

Abstract

Our goal is to design an IEEE 802.11e/AFDX hybrid network, then ensure the requirements of AFDX traffics in an IEEE 802.11e wireless communication medium. AFDX traffic constraints are temporal (a bounded delay and a bounded jitter). In this paper we present the maximum access delay of AFDX traffics in 801.11e wireless network with our scheduler named AWS (AFDX Wireless Scheduler). AWS is based on the classification of flows (priority class), on the retransmission management of frames, on the deadline of frames and on the maximal jitter. The performance evaluations show that we have a shorter maximum access delay for the most constraining flows with AWS compared to the Reference Scheduler (RS). The performance evaluations are performed under three scenarios (overloaded network, loaded network and moderately loaded network).

1. Introduction

The avionics systems are subject to very strict constraints to ensure working of hardware and software resources [1]. Several on-ground maintenances are performed on the on-board equipments in order to detect possible dysfunction. These tests may require the dismantling of equipments. This represents a cost in manpower and the aircraft will be ground for a longer duration. Thus, to facilitate the ground maintenance of equipments, our goal is to provide a wireless communication medium based on IEEE 802.11e standard, which allows us to perform the on-ground tests without the dismantling of equipments. For this, we design a hybrid network IEEE 802.11e/AFDX (Avionics Full DupleX switched ethernet). However, the use of the IEEE 802.11e WLAN [5] in an aeronautical context is a real challenge, due to the nature of the wireless communication radio, especially for: (1) the QoS, (2) non-deterministic access to wireless medium, (3) packet losses, (4) transmission errors and (5) share of channel.

In this paper we propose an access method based on the HCCA (HCF Controlled Channel Access) in order to: (1) reduce the access delay to wireless medium for flows with the most stringent requirements, (2) to control the jitter (3) and to avoid that frames reach their deadline. We use the HCCA access method because it offers us a larger flexibility of scheduling than the EDCA (Enhanced Distributed Channel Access) access method.

Note that, several enhancements have been proposed by the scientific committee to improve the performances of the Reference Scheduler (RS) and the admission control unit. In [6], they carry an estimate of the queues transmission length to determine the duration of TXOP for each QSTA (QoS STAtion), therefore there are no polling management and no jitter management. In [3], the *Wireless Timed Token Protocol* (WTTP) uses a list of *Round Robin* where nodes are placed only when they have frames at transmit. The reference duration of one round is equal to the smallest delays of the flows; in order to transmit all frames before their deadlines. In [7], the authors use the "Token Bucket" algorithm to regulate the input traffic for each station. From this assumption, they compute the maximal delay and propose an optimization of service interval. In [2], the *Real-Time HCCA* scheduler (RTH) is a periodic scheduling algorithm based both on the *Earliest Deadline First* (EDF) algorithm and on the *Stack Resource Policy* (SRP). Their aim is to design a scheduling algorithm that provides, to flows, fixed amounts over fixed periods. In [6], [3], [7], and [2] the authors have not taken into account the jitter and the additional traffic due to the retransmission of lost packets. In this paper we propose an algorithm based on flow classification (priority class), on deadline, on a jitter and on a policy for managing the retransmission of losted packets.

The paper is organized as follow; in Section II, we present the AFDX network. In Section III, we present the IEEE 802.11e standard. In section IV we present the topology of the IEEE 802.11e/AFDX hybrid network. Afterward, in section V, we propose our scheduling algorithm AWS for the IEEE 802.11e/AFDX hybrid wireless network and their performance evaluations. Finally, in Section VI, we give the conclusion.

2. AFDX network

AFDX is described specifically by Part 7 of the ARINC 664 Specification [1], as a special case of a profiled version of an IEEE 802.3 network utilizing IP addressing and related transport protocols. AFDX is faced to deterministic requirements which regard: (1) a low and bounded latency, (2) a low and bounded jitter, (3) an absence of data losses (4) and ensure the order of transmitted messages [4]. One example of AFDX network topology is illustrated in figure 1. The main function of the End-System (ES) is to provide services, which guarantee a secure and reliable data exchange to the application software. Its services must meet a guaranteed bandwidth and bounded latency, and a given maximal jitter. The ES uses the Virtual Link (VL) notion to regulate traffic data flows and exchanges Ethernet frames. At the output of the ES, each flow associated with a VL is characterized by two parameters: (1) Bandwidth Allocation Gap (BAG) and (2) Jitter. The BAG represents the minimum space between the first bits of two consecutive frames; if the frame is experience no jitter from the scheduler. BAG values should satisfy the following formula: BAG=2k [in ms], (k integer in range 0 to 7). A VL defines a logical unidirectional connection from one source ES to one or more destination ESs. Each VL has a dedicated maximum bandwidth (Lmax/BAG). Lmax is the maximum frame size. When transmitting ES has multiple VLs, it uses a scheduler to multiplex flows (VLs). End-Systems are connected by AFDX switch which is an Ethernet switch adapted to the aircraft network requirements.

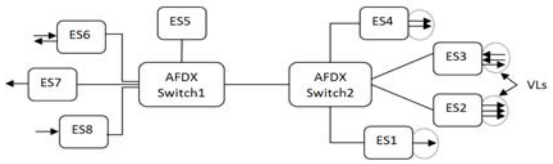


Figure 1. AFDX Network topology

3. IEEE 802.11e standard

The IEEE 802.11 standard characterizes a wireless local area network; it describes the MAC and Physical layers. Its version, IEEE 802.11e [5], offers support of QoS with the HCF (Hybrid Coordination Function) access mechanism. The HCF uses two access methods: EDCA (Enhanced Distributed Channel Access) and HCCA. Under HCF, the basic unit of allocation of the right to transmit onto the wireless medium (WM) is the TXOP *(Transmission Opportunity)*. A QSTA may transmit one or more data frames during a TXOP if there is sufficient duration. It is possible that no frame was transmitted during the TXOP. The HCCA mechanism uses a QoS-aware centralized coordinator, called a hybrid coordinator (HC). The HC is collocated with the Quality Access Point (QAP) and uses the higher priority of access to the wireless medium to initiate frame exchange sequences and to allocate TXOPs to itself and other QSTAs. Each polled QSTA by the HC receives a TXOP. The maximum value of TXOP is determined by the HC, which forwards it to the QSTA during the polling phase. A Reference Scheduling (RS) algorithm [5] is defined in order to determine the maximum value of TXOP (TXOPLimit) and the scheduled Service Interval (SI) of all QSTAs with admitted flows. The SI is the duration between two phases of polling and is chosen to be a submultiple of the beacon interval and the minimum of all maximum SIs of admitted flows. The scheduler calculates the TXOP duration as the maximum between (1) the time to transmit N_i frames at Physical Transmission Rate R_i and (2) the time to transmit one MSDU with the maximum size M_i at R_i, N_i is the number of MSDUs that arrived at mean data rate ρ during the scheduled SI.

$$TXOP_i = max((N_i \times L_i)/R_i + O, M_i/R_i + O) \quad (1)$$

L_i is the *Nominal MSDU Size* and *O* the Overheads in time units. The overheads include IFSs *(Inter-frame space)*, ACK frames and CF-Poll frames.

4. IEEE 802.11e/AFDX Hybrid network topology

To facilitate the ground maintenance of equipments, our goal is to provide a wireless communication medium based on IEEE 802.11e standard. This communication medium will allow us to connect us; either directly to the equipment, either from a switch AFDX to perform maintenance testing. In both cases, we use an QAP connected to an AFDX switch(see figure 2). At the QAP, we have a HC, which is responsible of scheduling the access to the wireless medium for VLs. In the wireless part, the QSTAs perform the role of End-Systems. However, as

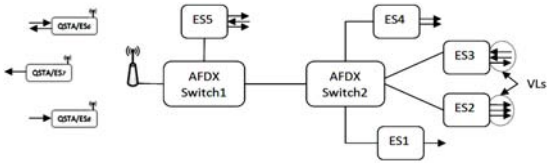


Figure 2. Hybrid network topology

we announced earlier, the wireless part of this hybrid network generates additional constraints that are related to the wireless medium. These constraints lead to a greater jitter and delay of frames and also reduce the bandwidth. A RS is provided to guarantee the QoS. However the RS is more adapted to CBR (Constant Bit Rate) traffic and its enhancements are oriented to multimedia applications. In this paper, we propose a HCCA scheduler, named AWS, that aims (1) to adapt the HCCA reference scheduler to the AFDX context and (2) to enhance it to avoid that frames reach their maximal jitter and maximal latency.

5. AFDX Wireless Scheduler (AWS)

To draw nearer to requirements of AFDX regarding the containment of errors related to behavioral dysfunctions of one or more VLs, we consider independently each VL. Thus in the following each VL is considered as a node in the network with periodic and sporadic traffic. Traffic is periodic with a *p* period, if it sends data at each *p* time interval. A periodic traffic is sporadic when it doesn't send a data for some periods. AWS aims to guarantee the transmission of all frames before their deadline (maximum delay and maximum jitter). It improves the reference scheduler: (1) by defining multiple priority levels which are based on the BAG values, (2) by using the deadline of frames to allow VLs with lowest priorities to transmit their frames, (3) by defining a retransmission management policy of frames, (4) by managing the sporadic aspect of VLs and (6) by a policy to control the jitter. AWS proposes a classification of VLs using the BAG values. A VL with a smaller BAG has the priority over a VL with a greater BAG. This priority policy between the VLs is designed in order to transmit at first the frames of the most constraining VLs. However, to avoid that the VLs with a low priority, have never transmit, AWS uses the deadline of frames. AWS set a threshold, which is based on the deadline, for each priority class. A frame of VL, which reached its threshold, receives the highest priority. AWS also set a management policy for the retransmission. The frame retransmission is no automatic. When a VL loses its frame, the HC first checks if it still has the highest priority. If so, it allows the transmission. Otherwise, the VL with high priority (closer to deadline) transmits its frame. To manage the sporadic aspect of VLs and avoid the unnecessary poll, the QSTAs use their data frames to communicate the state of their VLs to the HC, this by "piggybacking" the state of VLs in the frames of data. Each VL has four states: Idle, Active, Critical and Retransmission. A VL is in an Idle state when it is waiting for his next BAG or when it has reached its BAG and has no frame to transmit. VL is in an Active state when it has reached its BAG and has a frame to transmit. The Retransmission state is only for VLs having a transmission failure or a retransmission failure. The number of retransmission is limied by 7 for small frames and by 4 for large frames, according to IEEE 802.11. To control the jitter we assume that a VL is in a Critical state when it has reached its threshold (**BAG + jitter-max** - Δ). Δ is the critical period of VL. To avoid the un-used TXOPs, the HC scheduler schedules only VLs in states: Active, Retransmission, and Critical. Note that the HC have all information on VLs with the received TSPECs (Traffic Specification) of QSTAs.

5.1. Scheduling phase

After each end of transmission of a polled VL, the HC schedules the next TXOP on three steps at most:
Step 1: the HC updates the list of VLs in the critical state: it adds the new VLs reaching their critical state, removes VLs having reached their deadline and their packet is dropped.
Step 2: if the list of VLs in the critical state is empty after the update then HC will go to steps 3. Otherwise the HC allocates a TXOP to the VL with the highest priority in critical state. VL with the smallest deadline is the most priority.
Step 3: in this step, only the VLs in Active and Retransmission state are scheduled. The HC grants a TXOP to the VL having the smallest BAG. Nevertheless, if several VLs have the smallest BAG, the scheduler grants a TXOP to the VL that have the nearest deadline. And if several VLs have the smallest BAG and the same nearest deadline, then the HC follows the FIFO rule.

5.2. Simulations and results

The scenarios are based on a case of application issued by [4] (provided by Airbus). The left side of table 1 shows the distribution of VLs compared with BAG values. We note that the high values of BAG are most used in this configuration. In the right side of table 1, we have the distribution of VL compared with the frames size.

BAG(ms)	VL	-	frames size(bytes)	VL
2	2.03%	-	1-150	57%
4	4.06%	-	151-300	20.53%
8	7.93%	-	301-600	11.59%
16	14.43%	-	601-900	5.79%
32	23.27%	-	901-1200	1.22%
64	22.36%	-	901-1200	3.56%
128	25.91%	-	$\prec$ 1500	0.3%

Table 1. VLs distribution

In the following we only focus on the results of VLs with BAG values=2ms and 4ms because they have the most constraints for access delay . From the distribution table of VLs, we chose three scenarios. In the first scenario we have 30 VLs, that means TXOPs occupy 60 % of bandwidth (wireless area). The second scenario has 40 VLs (77% of bandwidth). In the last scenario we have 50 VLs, this represents 95% of bandwidth. For all scenarios, we have considered 4 QSTAs and one AP, we have chosen the maximum jitter equal to BAG/4 and the deadline equal to BAG plus the maximum jitter for each VL. The physical rate is 54Mbps (802.11a). The TXOP is performed as is defines in [5] equation (1). In following, we present and compare the perfomance mesures of RS and AWS, regarding the maximum access delay to the wireless medium.

Figures 3 and 4, show the maximum access delay to wireless medium for respectively VLs(BAG=2ms) and VLs(BAG=4ms), with BER=(10^{-5} and 10^{-4}), this for all scenarios. We note an increase of the maximum of access delay in according the network load. The maximum access delay of AWS is smaller than the maximum access delay of RS; we have at least a reduction of 50% of the maximum access delay with AWS compared to RS for VLs(BAG2ms) and for all scenorios. For VLs(BAG4ms),

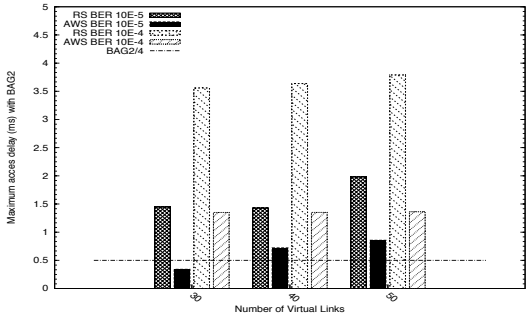


Figure 3. Maximum Access delay (BAG=2ms)

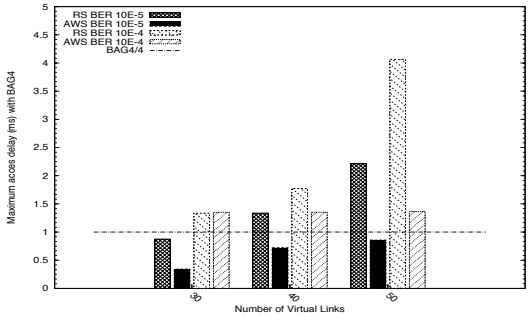


Figure 4. Maximum Access delay (BAG=4ms)

in a real case (BER=10^{-5}) we note that the maximum access delay is inferior to the deadline, whatever the scenario. This is not the case for RS. In some cases the maximum access delay of AWS exceeds the deadline, however we have a very large reduction of the dropped packet (reached deadline) with AWS compared to RS (figures 5 and 6). We note a gain of 24% for AWS in worst case situation (degraded wereless medium (BER=10^{-4}) and 95% of bandwidth occupancy by TXOPs). In a real case (BER=10^{-5}), dropped packets with AWS is always inferior to 3% for all scnarios versus 29% for RS.

6. Conclusion

The works presented in this paper are one part of study to design an IEEE 802.11e/AFDX hybrid network. The aim is to convey AFDX traffic on IEEE 802.11e wireless network. In this paper we have presented our HCCA scheduler named AWS to overcome the limitations of RS and reduce the access delay of the most constraining VLs. AWS scheduler is based on (1) the priority classes, (2) the deadline, (3) a policy for handle the retransmissions and also (4) the maximum jitter. The simulations results show that AWS scheduler are better access delay than the reference scheduler for the most constaining flows (figures 3 and 4). We note that we have a large reduction of the

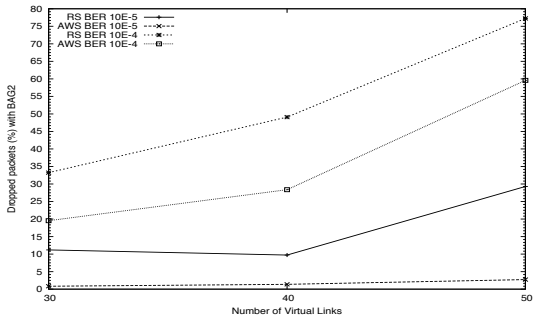


Figure 5. Dropped packets (BAG=2ms)

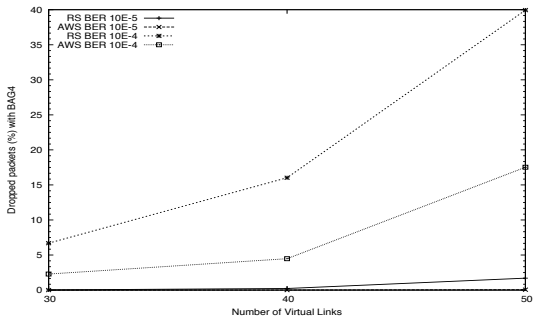


Figure 6. Dropped packets (BAG=4ms)

number of packets dropped with AWS. This involve a better compliance of deadline with AWS which is concretized by a gain of bandwidth. In our current work we have proposed a optimized version of AWS which aims to use the un-used bandwidth of TXOPs for the transmission of VLs in critical or retransmission state; in order to reduce the maximum access delay and the dropped packet number of AWS.

References

[1] I. Aeronautical Radio. *ARINC-SPECIFICATION-664. Aircraft Data Networks PART 7 AFDX*. New York, 2004.

[2] E. M. C.Cicconetti, L. Lenzini and G. Stea. Design and performance analysis of the real-time hcca scheduler for ieee 802.11e wlans. In *Comput. Netw.*, 2007.

[3] E. M. C.Cicconetti, L. Lenzini and G. Stea. An efficient cross layer scheduler for multimedia traffic in wireless local area networks with ieee 802.11e hcca. In *ACM Mob. Comput. and Commun. Reviews*, 2007.

[4] H. Charara. *Evaluation des performances temps reel de reseaux embarques avioniques*. PhD thesis, Institut National Polytechnique de Toulouse, 2004.

[5] IEEE-802.11e. *Specific requirements Part 11 : Wireless LAN medium access control (MAC) and physical layer (PHY) specification*. IEEE Computer Society, New York, 2007.

[6] Q. N. P. Ansel and T. Turletti. Fhcf :a simple and efficient cheduling scheme for ieee 802.11e wireless lan. In *Mobile Networks and Applications*, 2006.

[7] C. O. M. Y. Y.Higuchi, A.Foronda and Y. Okada. Delay guarantee and service interval optimization for hcca in ieee 802.11e wlan. In *IEEE Wireless Communications and Networking Conference*, 2007.

Worst Case End-to-End Delay Analysis of Switched Ethernet using Timed Automata

Muhammad Adnan, Jean-Luc Scharbarg, Jérôme Ermont, Christian Fraboul
Université de Toulouse/IRIT/ENSEEIHT/INPT 2, rue Camichel, 31000 Toulouse, France
{Muhammad.Adnan, Jean-Luc.Scharbarg, Jerome.Ermont, Christian.Fraboul}@enseeiht.fr

Abstract

Real-time applications require bounds on worst case end-to-end communication delays over switched Ethernet. Existing approaches determine these bounds pessimistically. The objective of this paper is to present an improved modeling approach using timed automata for calculation of exact worst case delays. This approach takes advantage of local scheduling of flows. Moreover, it can cope with larger network configurations than existing approaches based on timed automata, thanks to a port by port analysis which reduces the search space.

1. Introduction

A full duplex switched Ethernet network eliminates indeterminism in collision resolution mechanism of standard Ethernet CSMA/CD. In fact it shifts the indeterminism to the switch level where various flows can compete at each output port. Indeed, the waiting time of a frame in the output ports of the switches it crosses highly depends on the current number of pending frames in these ports; hence for real-time applications we need worst case delay analysis to determine the upper bound on end-to-end communication delays. This guarantee can be provided by Network Calculus [6, 4] however the results are pessimistic and do not present exact worst case delays. A model checking approach has been proposed [5]. It computes the exact worst case delay but makes no assumptions regarding the temporal relationship between the flows. This is true for flows coming from different source nodes except when a global time is present in the system (such as ProfiNet). This global time implementation leads to complex systems which are difficult to validate. When no global time exists, flows coming from a given node are still scheduled thanks to the local clock of this node as shown in Figure 5(a). Therefore, it is interesting to study the benefits of such partial scheduling of flows in the overall network.

The aim of this paper is to present an improved timed automata based modeling approach for finding exact worst case delays. This methodology also incorporates the scheduling at each node. Taking this partial synchronization of flows into account, we can improve overall delays and result in better network utilization. This approach can cope with larger networks, since the computation proceeds on a port by port basis, which reduces the search space.

The paper is organized as follows: Section 2 presents a brief overview of timed automata. Section 3 explains the improved modeling approach with an example and summarizes results on the example network. In section 4 the paper is concluded with directions for future research.

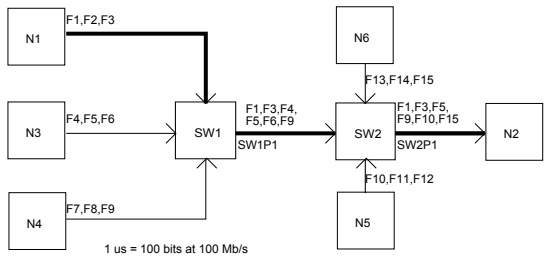
2. Overview of timed automata

Timed automata have been proposed first by Alur and Dill [2] in order to describe systems behavior with time. A timed automaton is a finite automaton with a set of clocks, *i.e.* real and positive variables increasing uniformly with time. Transitions labels can be: a guard, *i.e.* a condition on clock values, or actions, or updates which assign new value to clocks. The composition of timed automata is obtained by a synchronous product. Each action a executed by a first timed automaton corresponds to an action with the same name a executed in parallel by a second timed automaton. In other words, a transition which executes the action a can be fired only if another transition labeled a is possible. The two transitions are performed simultaneously. Thus communication uses the rendezvous mechanism.

Performing transitions requires no time. Conversely, time can run in nodes. Each node is labeled by an invariant, that is a boolean condition on clocks. A timed automata can be in a state only if the associated invariant is true.

Several extensions of timed automata have been proposed. One of these extensions is timed automata with shared integer variables. The principle consists in defining a set of integer variables which are shared by different timed automata. Consequently, the values of these variables can be consulted and updated by the different timed automata [8, 3].

A system modeled with timed automata can be verified using a reachability analysis which is performed by model-checking. It is done by verifying if a node is reached from the initial configuration. In the general



Flow	Size(us)	Period(ms)	Offset(ms)	SW1P1	SW2P1
F1	30	32	4	x	x
F2	20	32	8	-	-
F3	30	16	0	x	x
F4	50	32	0	x	-
F5	20	64	8	x	x
F6	30	128	24	x	-
F7	20	8	2	-	-
F8	10	4	0	-	-
F9	30	16	6	x	x
F10	50	64	24	-	x
F11	40	32	8	-	-
F12	30	16	0	-	-
F13	20	8	0	-	-
F14	30	16	4	-	-
F15	30	16	12	-	x

Figure 1. Example of switched Ethernet network configuration.

case, reachability analysis is undecidable on timed automata with shared integer variables. In the particular case where the shared variables are represented by nodes of a timed automata, the reachability analysis is decidable [8]. The approach considered in this paper is based on timed automata with shared integer variables which are represented by nodes of a timed automaton. The modeling of the switched Ethernet with timed automata, using UPPAAL [1], is presented in next section.

3. Timed Automata Based Modeling of Switched Ethernet Network

For this work we suppose following assumptions in full duplex switched Ethernet network composed of multiple nodes interconnected by switches using store and forward pattern. We assume that all the flows are strictly periodic and are statically defined and routed on the network including multicast flows. All nodes and switches support FIFO queuing. A flow comprises of strictly periodic frames sent from one source node to all destination nodes on one or several paths defined statically.

A model checking approach determines exact worst case delays but leads to combinatorial explosion thus restricting its use to only small networks [5]. To overcome this issue, a "divide and conquer" methodology is used in this paper. The idea is to calculate the exact worst-case delay on each output port for the path of given flow and then to propagate this delay in consecutive discrete steps starting from source node till the destination node.

The approach is illustrated by an example of switched Ethernet network shown in figure 1. This network consists of 6 nodes interconnected with two switches. We

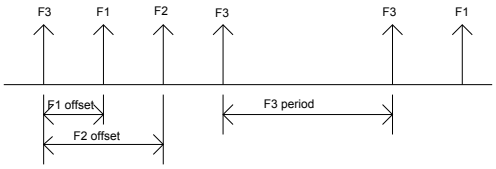


Figure 2. Offsets between flows of a node.

assume that all 15 data flows (F1 to F15) of this configuration are strictly periodic. Their maximum packet size and periods are given in figure 1. The scheduling implemented by each node is modeled by offsets associated to the flows, given in figure 1. We focus on F1 whose path is $\{$N1-SW1-SW2-N2$\}$ (bold line in figure 1). There are other flows originating from 'N1' and other end systems which pass through port 'SW1P1' and 'SW2P1' (indicated by 'x' in columns "SW1P1" and "SW2P1"). The calculation of exact upper bound on end-to-end delay of F1 is processed on a port by port manner; this delay is first computed at port "SW1P1" and the obtained value is then used to calculate the delay in port "SW2P1".

The description of the approach proceeds in 4 steps: a) Modeling of flows and their scheduling at one node. b) Modeling asynchronous behavior among all nodes. c) Modeling and computation of delay at first output port "SW1P1" and d) Modeling and computation of delay at subsequent output ports "SW2P1".

Modeling of flows and their scheduling at one node. As mentioned before, all flows are assumed strictly periodic with periods shown in figure 1. Hence their scheduling by given node is modeled by offsets. One flow is arbitrarily chosen as the first flow to generate a packet. Without loss of generality, we chose a flow with shortest period. In the example in figure 1, F3 is reference flow for "N1" and offsets of F1 and F2 are computed based on F3 shown in figure 2. Similarly F5, F8, F12 and F13 are chosen for "N3", "N4", "N5" and "N6" respectively. This leads to one timed automaton for each flow. While modeling the automata, all values are scaled by 10 μs.The automaton for F3 is depicted in figure 4(b). After the trigger of N1, we wait for the offset time to elapse (which in case of reference flow is zero) and then we go to a state which models jitter in the flow (for flows of originating node, this jitter is zero). Finally, we store the packet in FIFO queue and wait till the flow period is elapsed. This process is repeated periodically. Clock variable "x" in all TAs is a local variable.

Modeling asynchronous behavior among all nodes. The nodes are asynchronous with respect to each other. They can start transmitting flows at any time with reference to other nodes, as shown in Figure 3 where "StartN1" and "StartN3" can take any value between zero and largest period of the flows emitted by N1 and N3 respectively. This behavior is modeled in timed automaton

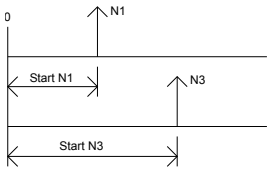


Figure 3. Asynchronous behavior of nodes.

in figure 4(a) representing N1. The state "n1" is reached after a delay that can take any value between zero and $32ms$ (largest period of any flow of N1 scaled by 10 μs). The model checking will test all possible values in this range.

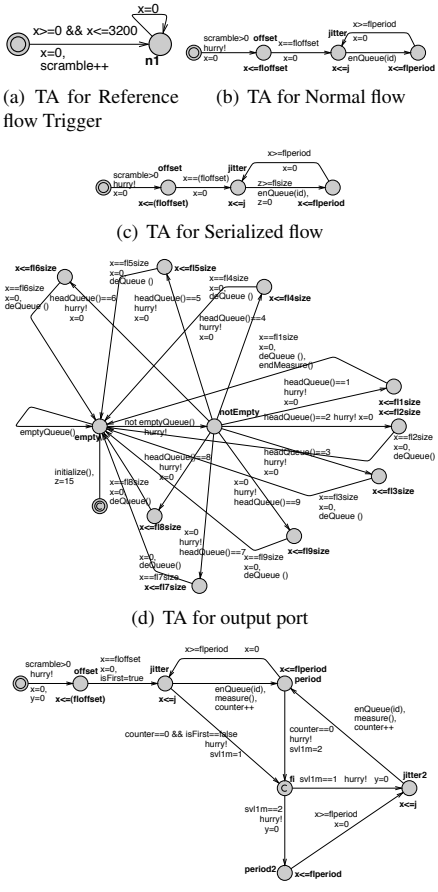
Modeling and computation of delay at first output port. The next step is the computation of upper delay at first output port $i.e$ "SW1P1". The output port is a FIFO queue (which serves packets in order of their arrival). Figure 4(d) shows timed automaton of an output port. It models the packet size, and the order among flows. For simplicity, we assume there is no latency within the switch but it can be easily added to the model. After initialization we wait in $empty$ state till we receive some packet. As soon as a packet is stored in queue, we check the ID of this packet representing the flow id by using function $headQueue()$ and go to corresponding state without elapsing any time thanks to urgent channel variable "hurry". The automaton has one location for each flow and stays in this location for the time equal to flow packet transmission time and then returns back to $empty$ state removing the packet from the queue. The FIFO queue is modeled as an array with maximum size equal to upper buffer bounds calculated by network calculus [7]. This ensures there will not be overflow in the queue. Functions $enQueue(id)$ and $deQueue()$ add and remove flow packets from the queue respectively.

The flow under analysis requires some additional states in it's model for measuring end-to-end delay. Figure 4(e) shows timed automaton of "measuring flow". The delay experienced by one packet of F1 is measured by local clock variable y. This clock is reset as soon as the packet is ready for transmission. It is measured when the packet has just been transmitted. The worst case delay is the smallest value x such that the value of clock y is always less than or equal to x. It is obtained by querying the timed automata using following CTL formula:

$$\text{A[] (F1m.fi imply F1m.y} \leq x)$$

The value x is initialized to the sure upper bound computed by the Network Calculus [7] and then decreased as long as the formula is verified. The delay for F1 in "SW1P1" is $110\mu s$.

Modeling and computation of delay at subsequent output ports. The jitter experienced by a flow at a given



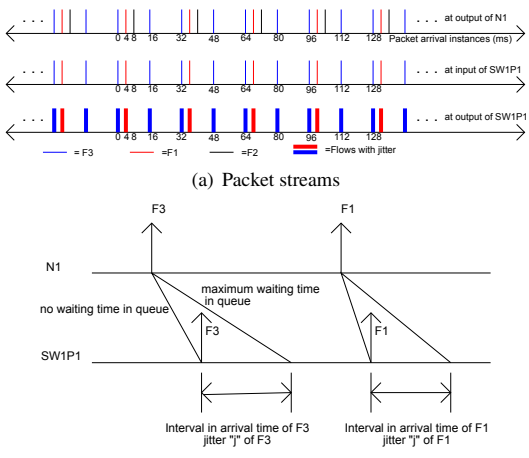
(a) TA for Reference flow Trigger (b) TA for Normal flow

(c) TA for Serialized flow

(d) TA for output port

(e) TA for Measuring flow

Figure 4. Timed Automata for flows.



(a) Packet streams

(b) Packet jitter at output port

Figure 5. Packet arrival instances and how a delay at output port appears as jitter.

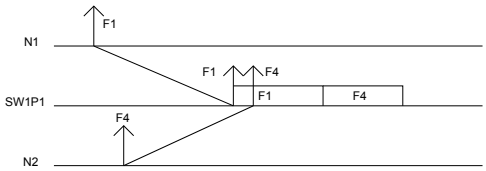


Figure 6. Serialization of flows after passing through an output port.

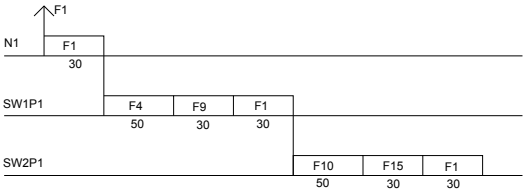


Figure 7. Worst case scenario for F1.

point in its path is clearly the maximum waiting time of a packet of this flow in the buffers of the output ports it has crossed as illustrated in figure 5(b). So the calculated delay at first port "SW1P1" is used as jitter j in timed automata models in second output port and so on. The above process is repeated for all output ports in the path of a given flow and whole network can be analyzed this way in port by port manner, albeit with a small addition to flow automata explained below.

In fact packets sharing a link are serialized, i.e they cannot be transmitted concurrently, as illustrated in figure 6, where the packet of F4 has to wait until the end of the transmission of the packet of F1. This serialization is modeled by the clock variable z in the timed automaton in figure 4(c). It ensures that the delay between the receptions of two consecutive packets 'P1' and 'P2' is at least the transmission time for 'P2'. This model is used for the example considered in this paper at "SW2P1" port for F1, F3, F5 and F9. The delay calculated at this port is 110 μs. The end to end delay is addition of all the delays on the path of the flow. In this example total delay of F1 is transmission time at N1+delay at SW1P1+delay at SW2P1 which equals 250μs. This worst case scenario is shown in figure 7.

Results. The table 1 represents the results of example network. All calculations were done on a machine with 3.3 GHz Intel Core 2 Duo processor having 4 GB RAM. In general for flows with no jitter, the computation time is about 5 seconds using approximately 15 MB of RAM but for flows with jitter, it can take up to 15 hours using approximately 4 GB of RAM.

Flow	Delay(us) at SW1P1	Delay(us) at SW2P1
F1	110	110
F3	110	110
F4	110	-
F5	80	100
F6	90	-
F9	110	110
F10	-	110
F15	-	110

Table 1. Exact worst-case delays.

4. Conclusion

In this paper, we have presented on a case study an improved timed automata based modeling technique for switched Ethernet network which integrates a more accurate model of flows. This approach is promising since it increases the size of the networks which can be analyzed, due to the port by port processing. Further work is still needed to apply this approach on larger configurations. It will probably be difficult to analyze a real industrial configuration (more than 1000 flows) but results on partial configurations can give valuable information on the pessimism of upper bounds obtained by Network Calculus. Moreover, a formal proof is needed in order to show that the port by port computation leads to an exact upper bound. This technique is not limited to switched Ethernet network but can cope with any similar network, provided flows are strictly periodic. It can also be used for analysis of different scheduling techniques and finding delay bounds in multi-processor networked architecture.

References

[1] http://www.uppaal.com.

[2] R. Alur and D. L. Dill. Theory of timed automata. *Theoritical Computer Science*, 126(2):183–235, 1994.

[3] A. B. Arjona. Vérification et synthèse de systèms temporisés par des méthods d'obervation et d'analyse paramétrique. *PhD thesis, Ecole Nationale Supérieur de l'Aéronautique et de l'Espace, Toulouse, France*, 1998.

[4] J.-Y. L. Boudec. Application of network calculus to guaranted service networks. *IEEE Transactions. Inf. Theory*, 44:1087–1096, May 1998.

[5] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul. Methods for bounding end-to-end delays on an afdx network. *ECRTS '06: Proceedings of the 18th Euromicro Conference on Real-Time Systems*, pages 193–202, 2006.

[6] R. Cruz. A calculus for network delay, part 1, part2. volume 37, pages 114–141, Jan 1991.

[7] C. Fraboul and F. Frances. Applicability of network calculus to the afdx. *Technical Report PBAR-JD-728.0821/2002*, 2002.

[8] K. G. Larsen, P. Pettersson, and W.Yi. Uppaal in a nuttshell. *International Journal on Software Tools for technology Transfer*, 1(1-2):134–152, 1997.

Towards a Behavioral Modeling of Real-Time Kernel in a Model-Driven Development Approach

Cédrick Lelionnais
ESEO-TRAME
4 rue Merlet de la Boulaye, BP 30926
F-49009 ANGERS cedex 01
cedrick.lelionnais@eseo.fr

Jérôme Delatour
ESEO-TRAME
4 rue Merlet de la Boulaye, BP 30926
F-49009 ANGERS cedex 01
jerome.delatour@eseo.fr

Abstract

One promising solution for meeting the demands of constantly-evolving Real-Time Embedded (RTE) software is the Model-Driven Development (MDD) approach, based on the principle of separating the description of an application from its platform-specific implementation.

Solutions exist for modeling the RTE software multitasking platform (real-time kernel) characteristics (services and properties offered by the platform). However, improvements of the behavioral description of those platforms have to be done.

A first experimentation, the formal modeling of the OSEK/VDX real-time kernel, has been carried out in order to explore the weaving of existing structural platform descriptions and behavioral aspects. A first set of leads and questions raised in this on-going work.

1. Introduction

Like other designers, Real-Time Embedded System (RTES) engineers are faced with the challenge of developing more complex, higher quality systems at lower costs, with quicker development cycles. Within this context, reuse, maintainability and portability have become major issues in RTES design processes. The Real-Time Operating Systems (RTOS) provided a response to these problems by ensuring independence with regard to hardware platforms. Efforts have been made in order to standardize the Application Programming Interfaces (APIs) of RTOS [1] [2] [3]. To a certain extent, part of the applications could be reuse and migrate from one RTOS to another. But this task is not automatic between all the RTOS yet.

A new approach, based on MDD, addresses solutions to this problem. Indeed, MDD places the model paradigm and the use of model transformations at the center of the development process. MDD also promotes to separate concerns that are platform-specific (dependent of an execution technology) from those that are platform-independent. The Model Driven Architecture (MDA) [13] approach involves a design process where a Platform-Independent Model (PIM) (a description of the application without platform consideration) is transformed into a Platform-Specific Model (PSM), given a Platform Description Model (PDM) (a description of the platform).

Nowadays, the proposals for modeling PDM relies on static descriptions (list of services and properties offered by the platform), without taking into account the behavioral aspect (the operational semantic) of the platform.

In RTES, the correct behavior of the applications depends not only on the correctness of the results of computations but also on the times at which these results are produced. As the behavior of a RTES is dependent of the behavior of its execution platform, it is therefore important that the PDM's behavior (used in RTES) could be described.

In this paper, we seek to include the behavior into PDM description. Firstly, explanations on the modeling of RTE software platform are given. Secondly, a first experimentation on the integration of the behavior in the RTE software platform modeling is presented. This experimentation targets the OSEK/VDX real-time kernel and its translation into Petri Net (a formal language for describing the behavior). Finally, we conclude on this experiment.

2. RTE Software Platform Modeling

This section is divided into three sub-sections: the first is dedicated to the description of RTE software platforms, the second to the use of platform modeling and the third to explain the interest of formalizing the behavior.

2.1. What characterizes RTE Software Platforms?

In the context of MDD, an execution platform is regarded as “a set of subsystems and technologies” that provides the capabilities needed to support the execution

of a software application [5]. Their modeling goes through different characterizations (see [4] for a detailed presentation) :

– **Resources** represent the concepts offered by the platform. For instance, in a RTOS, the tasks, the semaphores, the mailboxes are considered as resources.

– **Services** are offered by the resources and caught at APIs's level. For example, semaphore's services are the “take” and “release” primitives.

– **Use rules** of the platforms describe the way the resources and the services could be used.

– **Life cycle** describes the observable behavior of the resources and the services.

Based on these characterizations, several languages (more exactly Metamodels) for modeling the platform exist : In the MARTE's UML profile [7], the “Software Resource Modeling” package [6] has been defined for modeling the RTOS APIs. In the Domain Specific Modeling Language (DSML) approach, the Real Time Embedded Platform Modeling Language (RTEPML [8]) enables to describe a software platform.

However only the structural aspect (Resources & Services) is represented. It is allowed to model the APIs of RTOS, but neither the use rules nor the observable behavior.

2.2. Usefulness of RTOS Modeling

Modeling a RTOS as a PDM seems interesting. It enables to separate the application (PIM), and the RTOS's technology (PDM) of a system. Therefore, one could generate, from the same PIM, different PSMs (each PSM targeting a particular RTOS). Different approaches of transformation exist [8] but, one, called the explicit approach, allows to have a generic transformation parametrized by the PDM. In that approach, the PDM is described in a language. In MDD terms, the PDM conforms to a metamodel (see Figure 1).

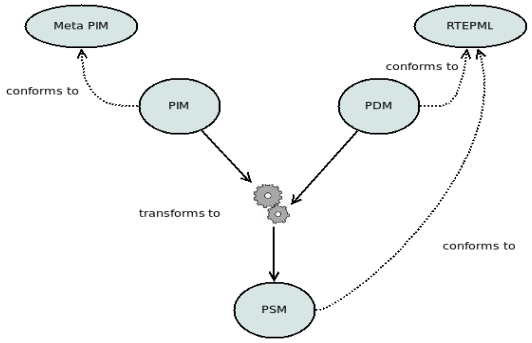


Figure 1. PIM to PSM Transformation.

A trivial example is here given to present a PIM/PDM/PSM description (see Figure 2). It is a robot's application (PIM) which comprises two concurrent tasks, one for control and one for display. These tasks have different priorities and share the same semaphore. This application will run on an OSEK/VDX real-time kernel [2].

We choose to use the RTEPML approach to represent the PDM and PSM. The PDM describes the notion of task, semaphore, their services and some typed elements (here integers). For the sake of simplicity, only a sub-part of the OSEK/VDX PDM is given. The PSM (also conforms to RTEPLM) contains the task instances (here two OSEK/VDX basic task) and a semaphore. The PSM elements are typed by those of the PDM.

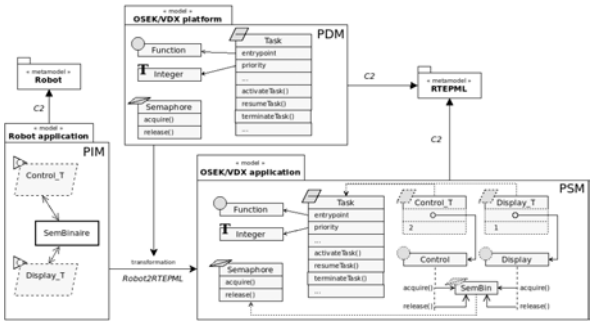


Figure 2. Explicit Description.

Nevertheless this description does not represent the behavior of the PDM. For instance, the token semaphore's properties change (from locked to unlocked) could not be represented.

2.3. A need of behavioral modeling

In the OSEK/VDX norm, the operational semantics (what we called in this article the PDM's behavior) of this RTOS is described by state-machines and textual description. For instance, in figure 3, part of the behavior of the OSEK/VDX “basic task” concept is given. It models the effect of the call of task's services on the task's states. A suspended task could not be executed by the scheduler. In order to be executed, a suspended task must be activated (call of the “activate” service) and then chosen by the scheduler (call of the “resume” service).

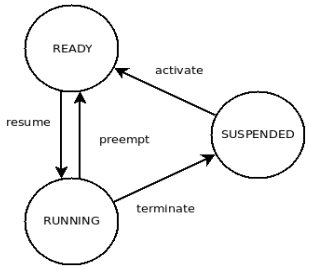


Figure 3. OSEK/VDX basic task behavior.

Without a formal description (done by a formal language) of the behavior, there is still the risk of ambiguities or inconsistencies in the description of the operational semantics of the execution platform.

Formal behavioral modeling would also offer the possibility to verify and validate the PDM and PSM

behavior (respect of temporal conditions, search of deadlocks...).

3. Towards a Formal Behavioral Modeling

In our research team, we already developed a PIM to PSM transformation with a RTEPML description of the OSEK/VDX RTOS. This transformation prototype, called “Robots2Osek”, generates C code from a PIM description (a description of robot's application, see figure 2). This generated C code is then executed on the TRAMPOLINE RTOS [12] (an OSEK/VDX RTOS). This transformation produces an intermediate model, a PSM where all the PIM, PDM and PSM elements are available. As already described, the RTEPML metamodel used in this transformation did not describe the behavior of the PDM. In order to add a behavior description in RTEPML, we carried out a first experimentation.

3.1. Description of the experimentation

For including behavior in RTEPML description, we choose to use the Petri Net (PN) language [9], mainly for its ability to represent concurrent activities, and also for the availability of Verification and Validation tools like ROMEO [10].

We first start a translation of the behavior of each concept of the OSEK/VDX (basic task, extended task, event, message box,..) in PN. Figure 4 depicts one of this translation (the OSEK/VDX concept of “basic task”). This translation relies on the expressiveness of a scheduling in PN [14].

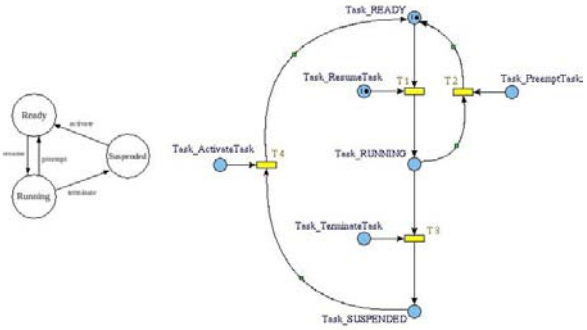


Figure 4. Behavioral pattern of an OSEK/VDX “basic task” in PN.

We named “behavioral pattern” this kind of PN fragment. In the figure 4, the PN's places named “Task_READY” and “Task_ResumeTask” are marked by a token, which means that the transition “T1” is going to be fired. At the next sequence, the place “Task_RUNNING” will be marked, meaning a change of the task's state. Also the behavioral pattern translated makes “source” places appear. These places correspond to the call services of a task and will be used for composing the final PSM description. For example the place “Task_ActivateTask” could be merged with one of another behavioral pattern.

In order to verify the translation correctness of our behavioral patterns, we choose to develop a PSM to Petri Net transformation. The idea was to simulate the obtained PN in ROMEO and compare this simulation with the C code execution of the same PSM obtained by our existing “Robots2Osek” transformation.

3.2. Transformation prototype developed

We then developed a transformation prototype (see Figure 5) with the transformation tool Kermeta [11]. The translation of the task (“basic” and “extended”), the semaphore, the alarm and the event have been implemented.

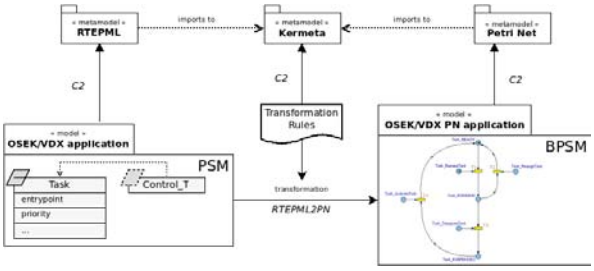


Figure 5. Transformation prototype.

Figure 6 depicts the generated PN of the translation of the robot's application example given in figure 2.

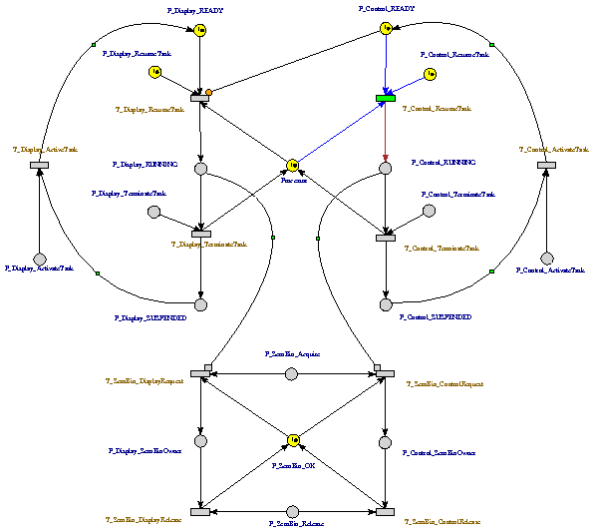


Figure 6. Generated PN.

Two behavioral patterns could be observed :

– The sequential execution of task (see Figure 4), one for the control task, one for the display task.

– The semaphore sharing.

Notice that the higher priority of the control task is modeled by an inhibitive arc. The merging of places and transitions is automatically generated by the transformation.

3.3. Feedback on work realized

The presented experimentation is a first step in our on-going work. We think that it demonstrates the feasibility and the interest of the MDD approach. The experimentation was carried out in less than 6 months. Most of the time was spent in the study of the behavioral pattern rather than in the transformation development techniques (thanks to the KERMETA tools and the RTEPML metamodel availability).

The transformation developed is obviously not generic and is not reusable for other RTOS. The behavioral patterns were buried into the transformation, rather than added into the platform description (the RTEPML metamodel). Works have still to be done in order to check the correctness of our behavioral pattern. The comparison between the simulation of the generated PN and the execution of the C code (generated by the "Robots2Osek" transformation) is far to be terminated. Only some basic examples have been successfully compared. Also the PN readability is rapidly difficult : with more complex synchronization mechanisms (message box or alarm) and more complex application examples, the number of transitions and places grows rapidly. However, we have now a better understanding of the PDM mechanism to translate and also assumptions on the required expressiveness of the formal language to use.

4. Conclusion and Future Prospects

The MDD approach promotes to separate concerns that are platform-specific (dependent of an execution technology) from those that are platform-independent. Therefore, transformation could be defined in order to automatically generate the further into the former. But, a description of the platform characteristics (a PDM) has to be done. For RTE platforms, the operational semantics of the platform has to be captured.

In the formalisms available to describe PDM, none could model, in a formal language, the operational semantics.

A prototype transformation has been developed in order to investigate the inclusion of the operational semantics in current PDM descriptions. It targets the OSEK/VDX RTOS and generates Petri Nets.

This experimentation shows the interest and the feasibility of the MDD approach. However, it has to be reproduced on different RTOS (such as VxWorks, ARINC 653, RT-Linux) in order to raise common behavioral patterns for the translation. Moreover, this pattern will have to be included in a PDM formalism (leading to an extension of the syntax of metamodels such as RTEMPL or UML-MARTE profil).

Another point is the consideration of the formal language to use. Although the expressiveness of Time Petri Net [14] seems adequate (despite combinatorial explosion risk), other formal language families have to be studied.

References

[1] The Open Group Base Specifications, "Portable Operating System Interface (POSIX)", ANSI/IEEE Std 1003.1, 2004.

[2] OSEK/VDX Group, "OSEK/VDX OS specification", Version 2.2.3, *http://portal.osek-vdx.org/files/pdf/specs/os223.pdf*, 2005.

[3] Airlines electronic engineering committee, "Avionics Application Software Standard Interface, ARINC Specification 653-1", Aeronautical radio, INC., Annapolis, Maryland, USA. October 2003.

[4] F. Thomas, J. Delatour, F. Terrier, S. Gerard, "Towards a Framework for Explicit Platform Based Transformations", 11th IEEE symposium ISORC, Florida, may 2008

[5] S. Beydeda and V. Gruhn, "Model-Driven Software Development, Volume II" *A Generalized Notion of Platforms for Model Driven Development,* 2005.

[6] F. Thomas, S. Gérard, J. Delatour, F. Terrier, "Software Multitasking Resource Modelling", Lecture Notes in Electrical Engineering (LNEE), issue on "*Embedded Systems Specification and Design Languages*", Vol. 10, Springer-Verlag, 2008.

[7] Object Management Group, "UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)", *RFP* 2005.,*OMG document: realtime/05-02-06*.

[8] M. Brun, J. Delatour, Y. Trinquet, F. Thomas, S. Gérard, "Étude comparative pour la modélisation de plates-formes d'exécution", TSI, numéro spécial "Ingénierie dirigée par les modèles", vol 29, 2010.

[9] C.A. Petri, "Kommunikation mit Automaten", PhD Report, University of Bonn (Germany), 1962.

[10] G. Gardey, D. Lime, M. Magnin, O.(H.) Roux, "Roméo: A tool for analyzing time Petri nets". In 17th International Conference on Computer Aided Verification (CAV'05), Lecture Notes in Computer Science, Edinburgh, Scotland, UK, July 2005. Springer.

[11] Triskell project (IRISA): "The metamodeling language kermeta". *http://www.kermeta.org* (2006)

[12] J-L. Bechennec, M. Briday, S. Faucou, Y. Trinquet, "Trampoline An Open Source Implementation of the OSEK/VDX RTOS Specification", Emerging Technologies and Factory Automation – ETFA'06, 2006.

[13] OMG, "MDA Guide", V1.0.1. OMG Document: omg/03-06-01, 2003.

[14] D. Lime and O. (H.) Roux. "Expressiveness and analysis of scheduling extended time Petri nets". In 5th IFAC International Conference on Fieldbus Systems and their Applications, (FET'03), pages 193-202, Aveiro, Portugal, July 2003. Elsevier Science. Copyright Elsevier Science.

A Multi-Class Architecture for a Differentiated Execution of Real-Time Transactions

Sami Limam, Leila Baccouche
Riadi-GDL Laboratory
University of
La Manouba, Tunisia
Limamsami@yahoo.fr
Leila.Baccouche@free.fr

Bruno Sadeg
Laboratoire LITIS
UFR des Sciences et Techniques
25 rue Philippe Lebon, BP 540,
F-76058 Le Havre, France
Bruno.Sadeg@univ-lehavre.fr

Henda Ben Ghezala
Riadi-GDL Laboratory
University of
la Manouba, Tunisia
Henda.BG@cck.rnu.tn

Abstract

Current applications such as stock markets, e-business, multimedia and telecommunications require more and more real-time services. Such systems deal with large quantities of data, and transactions have temporal constraints. Consequently, when they are increasingly solicited, they can quickly become overloaded, leading some transactions to miss their deadlines. To deal with this problem, we propose, in this paper, a multi-class architecture based on the importance of transactions that provide differentiation of services during their execution. Our work is based on the (m,k)-firm model when we serve the transactions classes. We show that this approach allows to provide different levels of quality of service (QoS), enhancing then the management of real-time transactions.

1. Introduction

Recently, the demand for real-time services has increased considerably in many applications such as multimedia applications (video conferencing, video on demand services ...), Web services and e-business. In addition, these applications reflect an important requirement in data management. Therefore, real-time database systems (RTDBS) have become the proper support to the implementation of these applications. In these applications, it is important to obtain complete and accurate results before the deadline, while using fresh data. However, as the users requests remain unforeseeable, the RTDBS can become overloaded leading to the system inability to meet deadlines and to control the data freshness.

To handle overload situations, several approaches based on QoS have been proposed either to execute the transaction that provide a positive value to the system [8, 6], or to divide the transaction into an obligatory part and an optional part [9].

Consequently, in a RTDBS, it's often interesting to categorize transactions into classes. An appropriate architecture for the differentiation between the classes of transactions is required. The RTDBS must incorporate an appropriate scheduling algorithm, which aims to guarantee QoS metrics for each class.

In this paper, we propose such an architecture which allows a differentiation of services between the transactions classes. To this end, we exploit the (m,k)-firm transactions model and we propose a scheduling algorithm based on this model.

The paper is organized as follows : In Section 2, we present different existing transactions models in a RTDBS and a discussion about them. Then we choose one, and we argue our choice.

In Section 3, we present the architecture we propose to manage transactions classes. Section 4 is devoted to the presentation of the simulations we conducted to validate our model, as well as to comments on the results obtained. The simulation platform used is called RTDS (Real-Time Database Simulator) which we have developed in our laboratory. The simulations results show the effectiveness of the proposed architecture and its capability to support overload situations and to guarantee QoS for each transaction class. Finally, we conclude the paper.

2. Model of transactions

In a RTDBS, we distinguish between two types of transactions:

- Update Transactions: they are periodic and have to refresh regularly the database by updating data.
- User transactions: they read/write non real-time data and/or only read real-time data. As their arrival in the system is unpredictable, they may cause overload situations. That is why imprecise transactions models are often used.

In this paper, we assume that transactions have firm deadlines, i.e. a transaction which misses its deadline becomes useless and is aborted. In addition, we exploit the imprecision model of transactions in overload situations. A transaction consists of a mandatory part and an optional part. The mandatory part must be executed before the transaction deadline. Optional part is composed of sub-transactions which are executed if enough time remains before deadline. The greater the number of optional sub-transactions executed, the better the result is, i.e. the quality of service of the result is enhanced.

Three main models exist that use imprecision of transactions: (i) Milestone model [1], (ii) (m,k)-firm model [5], and (iii) Kang's model [10]. We have chosen the (m,k)-firm model as the transaction is divided into k sub-transactions (m are mandatory and (k-m) are optional) and we have to execute only mandatory parts.

As all mandatory parts have to be executed, we use this model by assuming that m parameter is equal to 1, i.e. we consider only one mandatory sub-transaction.

3. System architecture

3.1. Queue's Model

We adopt a multi queues model with a single server. Indeed, in our model of transactions, we have different types of transactions, executed by only one processor.

We begin to determine the number of queues needed by the architecture.

We have defined a parameter, called *Importance* which will differentiate the types of transactions. We define three levels of importance:

- High: it qualifies update transactions.
- Medium: user transactions performing write operations.
- Low: user transactions that perform read operations.

We then define three queues corresponding to the three levels of importance. As a transaction consists of a mandatory sub-transaction and several optional sub-transactions, we have mandatory parts and optional parts, in each queue. As transaction classes need to be separated to allow service differentiation, each queue is divided into two queues: one is used to contain the mandatory parts and the other to contain the optional parts (cf. Fig. 1).

We then obtain a model with five queues:

- One for update transactions.
- Two for the high importance user transactions: one containing mandatory sub-transactions, the other one containing optional sub-transactions.
- Two for the low importance user transactions: one containing mandatory sub-transactions, the other one containing optional sub-transactions.

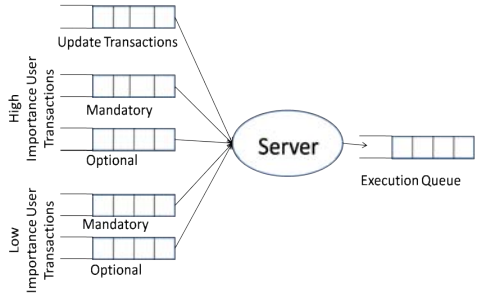


Figure 1. Queue model.

We serve these queues by adopting an algorithm based on the (m,k)-firm model, which provides a quality of service according to each class of transactions.

3.2. DBP_CC : an algorithm to serve queues

Our problem is to schedule transactions inserted in the different queues. We thus choose DBP algorithm [7] already proposed to schedule network packets on several queues. This algorithm is adapted to the real-time context and allows to guarantee QoS. However, it does not include a concurrency controller.

For this purpose, we proposed DBP_CC algorithm (Distance Based Priority and Concurrency Control) [2, 3], which is an adaptation of DBP algorithm [7](Distance Based Priority) to the RTDBS context.

DBP_CC is a dynamic algorithm issued from the (m,k)-firm model. The algorithm we propose provides several levels of QoS described by various couples (m,k) specified by the database administrator for each class of tasks.

DBP_CC saves the execution history of each queue in a structure called k-sequence, which is a sequence of k bits updated after the execution of the task (1 indicates that deadline is met, and 0 indicates the deadline is missed). Based on this history, DBP_CC computes a priority for each queue.

$$Priority\_DBP = k - l(m, S) + 1$$

Where l(m,s) is the position leaving from the right of the m^{th} success (1) in the k-sequence, s (the state of the queue).

DBP_CC extracts the transaction in the head of the more prioritized queue. In addition, if this transaction is in conflict of resources with the set of transactions in execution, we choose to extract the next one.

4. Transactions processing steps

In our architecture, a transaction follows three steps: insertion, extraction and execution:

4.1. Insertion and Extraction

As soon as it has arrived in the system, a transaction is inserted in the queue according to its importance. In each

Parameter	Value
Number of ressources	100
Number of temporal ressources	20
Number of non real-time resources	80
Load of update transactions	40%
Execution time of periodic transactions	30 to 150 ms
Execution time of a user transaction	70 to 150 ms
Number of mandatory sub-transactions in a user transaction	1
Number of optional sub-transactions	1 to 4
Number of resources used per sub-transaction	1

Figure 2. Simulation parameters.

queue, transactions are ordered according to EDF policy (Earliest Deadline First), i.e. the closest is a transaction to its deadline, the highest is its priority.

Based on DBP_CC, the next transaction to be extracted is from the nearest queue to the dynamic failure state.

4.2. Execution

Once inserted in the execution queue, transactions are executed using the Round Robin policy, i.e. each transaction is executed during a quantum of time. During the execution, the concurrency control algorithm 2PL-HP (2-Phase-Locking-High Priority) is applied to respect the isolation property.

However, the Round Robin policy doe's not distinguish between transactions types. Our goal through the proposed architecture is firstly, to differentiate transactions according to their importance and type (mandatory or optional), and secondly, to take into account service differentiation also during transaction execution so the quantum of time allowed to sub-transaction execution should depend on its importance.

5. Simulations and results

5.1. Simulation Context

Simulation of the protocols developed for RTDBS requires a platform which respects transactions ACID (Atomicity, Consistency, Isolation, Durability) properties and which offers adequate mechanisms to support time constraints.

RTDS [4] is a discrete-event simulator written in Java, designed by our research team to simulate real-time database behavior.

The following table summarize the simulation parameters :

5.2. Simulation results

We carried out three kinds of simulations:

1. In the first simulation, we show the impact of the system load on the system performances. In the simulation, we have built three queues, each queue holding a class of transactions (mandatory and optional parts of a transaction are put in the same queue).

2. In the second simulation, we show the impact of the variation of the number of queues. In this simulation, we separate the mandatory parts from the optional parts.

3. In the last simulation, we analyze the impact of allocating more than one quantum of time to the execution of update transactions.

5.2.1 Simulation 1: Variation of the system load

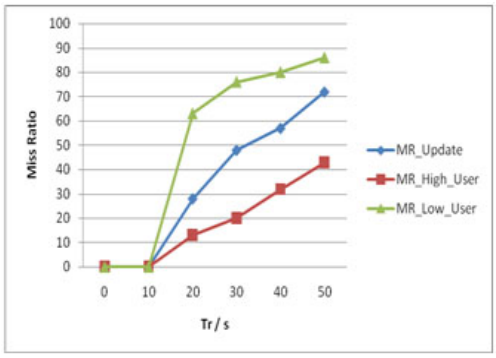


Figure 3. Transaction Miss Ratio when using (m,k)-firm model with three queues in the system.

In Figure 3, MR_Update represents miss ratio of update transactions, MR_High_User represents miss ratio of high importance user transactions, and MR_Low_User represents miss ratio of low importance user transactions. In the simulation, we have three classes of transactions materialized by three queues: High, Medium and Low with respectively the following (m,k)-firm constraints: (20,20), (14,20) and (2,20). Figure 3 shows the variation of the transactions miss ratio according to the load of the system when a transaction consists of one mandatory sub-transaction and an optional part (one or more sub-transactions). We notice that the transaction miss ratio for the periodic transactions class is greater than that of user transactions (whatever the importance criterion). This is due to the fact that the execution time of an update transaction (between 30ms and 150ms) is greater than that of a mandatory user sub-transaction ($\geq$ 20 ms).

5.2.2 Simulation 2: Variation of the number of queues

In Figure 4, transactions are dispatched in 5 queues. When we separate mandatory parts of user transactions from op-

tional parts, we can apply a different (m,k)-firm constraint for each queue. In Figure 3, we notice that the miss ratio of update transactions decreases without affecting the global Miss ratio (mandatory and optional) of user transactions.

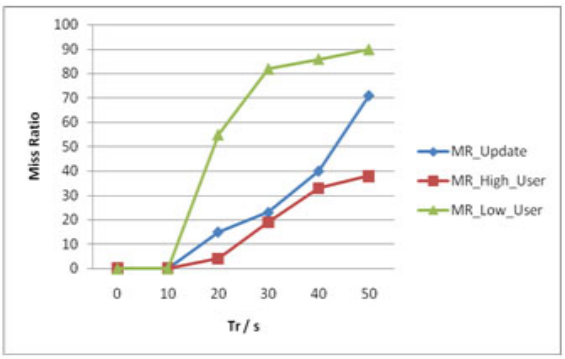


Figure 4. Transaction Miss Ratio when system queues are 5.

5.2.3 Simulation 3: Allocating more time to the execution of update transactions.

In Figure 5, execution of update transactions is allocated much more time than to user transaction (three time quanta). Consequently, the miss ratio of update transactions decreases. We notice that if we differentiate the service of transactions during the extraction phase and its execution phases, we obtain the best performances.

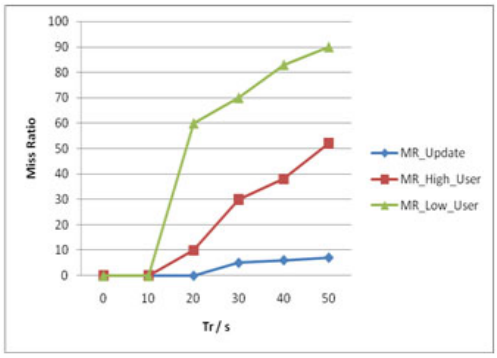


Figure 5. Miss Ratio when we grant more time to the execution of update transactions.

6. Conclusion

In RTDBS, execution of transactions before their deadlines leads often to overload situations, during which the system performances are degraded. Based on (m,k)-firm model, we proposed, in this paper, an approach to control this degradation for each class of transactions. Results of the simulations we have carried out have shown that when using an appropriate model of queues and when choosing efficiently the parameters of the (m,k)-firm model, we are able to differentiate between transactions classes and to give more execution time to more important transactions. A transaction consists of a mandatory part and several optional parts. We dispatch each transaction class between two queues: one containing the mandatory parts and the other contains optional parts. Then, we choose an (m,k)-firm constraint for each queue and we serve queues by applying the DBP_CC algorithm. Finally, we allocate more time to the execution of important transactions. In this way, we can control the QoS of real-time applications.

References

[1] M. Amirijoo, J. Hansson, and S. Son. Specification and management of qos in real-time databases supporting imprecise computations. *IEEE Transactions on Computers*, 2004.

[2] L. Baccouche. An overview of moa, a multi-class overload architecture for real-time database systems: framework and algorithms. *in Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (Dubai/Sharjah, March)*, pages 756–763, 2006.

[3] L. Baccouche and S. Limam. (m,k)-firm scheduling of real-time transactions with concurrency control. *in Proceedings of the second international conference on systems, Guadeloupe, French Carribean*, pages 14–21, 2007.

[4] L. Baccouche and S. Limam. Rtds: A component and aspect-based real-time database system simulator. *in Proceedings of ESM08, Le Havre*, pages 551–555, 2008.

[5] G. Bernat, A. Burns, and A. Llamos. Weakly hard real-time systems. *IEEE Transactions on Computers*, pages 308–321, 1999.

[6] A. Bestavros and S. Nagy. Value-cognizant admission control for rtdb systems. *In proceedings of the 17th Real-Time Systems Symposium (RTSS'96)*, pages 230–239, 1996.

[7] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m,k)-firm deadlines. *IEEE Transactions on Computers, 44(12)*, pages 1443–1451, 1995.

[8] J. Hansson and S. Andler. Value-driven multi-class overload management. *In Proceedings of the 6th Conference on Real-Time Computing Systems and Applications (RTCSA'99)*, 1999.

[9] J. Haubert, B. Sadeg, and L. Amanton. Relaxing the real-time constraints in distributed real-time management systems. *Proc. of the 10th Intl. Real-Time Computing Systems and Applications Conf. (RTCSA'04), Gothenborg, Sweden, Springer-Verlag ed.*, 2004.

[10] K. Kang, S. Son, and J. Stankovic. Service differentiation in real-time main memory databases. *In Proceedings of the 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, (IEEE ISORC), Washington D.C*, pages 119–128, 2002.