PEGASE – a robust and efficient tool for worst-case network traversal time evaluation on AFDX[1]

Authors :
Marc Boyer – ONERA, The French Aerospace Lab – F31055 Toulouse
Jörn Migge – RealTime-at-Work – F54600 Villers-lès-Nancy
Marc Fumey – Thales Avionics – F31100 Toulouse
Date : 06/06/2011
Document id : technical report based a paper to be presented at SAE Aerotech 2011.
Contact author: Marc.Boyer@onera.fr

[1] This work has been partially funded by French ANR agency under project id ANR-09-SEGI-009.

1 Introduction

Avionics functions are nowadays implemented by real-time applications, running on shared computers and communicating with each other. The real-time constraints typically associated with local applicative tasks now extend to the communication networks between sensors/actuators and computers, and between the computers themselves. Once a communication medium is shared (like an AFDX backbone), the time between sending and receiving a message not only depends on the technological constraints, but mainly on the interactions between the different streams of data sharing the medium.

It is therefore necessary to have techniques and tools to guarantee, in addition to local scheduling requirements, the worst case traversal time of the network (WCTT) and thus ensure a correct global real-time behavior of the distributed applications/functions. The network calculus is an active research area [Chang00, LeBoudec01] based on the (min,+) algebra [Baccelli92], that has been developed to compute such guaranteed bounds. They already exists several academics implementations (see [Boyer10b] for an overview), but no up to date industrial implementation. To address this need, the PEGASE project gathers academics and industrial partners to provide a high quality, efficient and safe tool for the design of avionic networks using worst case performance guarantees.

The PEGASE software is an up-to-date software in the sense that it integrates the latest results of the theories, in tight cooperation with academics researchers. The PEGASE software is also a safe tool: it relies on a strong mathematical background, its algorithms are described in a formal document, and it shares unitary tests with some academics tools. The PEGASE software is a usable tool: in particular, it has been designed from the requirements expressed by embedded network engineers. Finally, the PEGASE software is an efficient tool: it provides bounds close to the actual real worst-case, avoiding over provisioning of resources.

This paper presents the mathematical background of the tool, its architecture and some first results on realistic case studies.

1.1 The AFDX network

The AFDX technology (Avionics Full-Duplex Switched Ethernet, ARINC 664 standard part 7 [AFDX]) is an embedded network that is based on the Ethernet technology. If standard Ethernet offers a large bandwidth, it su ffers from well-know indeterminism that rules out its use in real-time systems. On AFDX, each node, termed an end-system, is connected to a communication switch with full-duplex links: there cannot be any collision on the links, but indeterminism may still arise from the waiting times in the queues of the AFDX switches. However, as on an AFDX network it is imposed that the amount of frames to be sent by the nodes over a given time interval is always bounded, it becomes possible to compute the worst-case transmission delay for a frame from the source to the destination(s). This delay, also called the Worst-Case Traversal Time, must be upper-bounded as explained in next paragraph.

1.2 Worst-Case Traversal Time (WCTT) evaluation: an industrial requirement

With the increasing amount of critical data exchanged with real-time constraints in on-board aerospace systems, the computation of tight upper bounds on network traversal times is

becoming a real industrial need. The reason is twofold. First, a tight and safe dimensioning of the hardware and software architecture is necessary. Second, it is required in the certification process to convince the certification authorities that the real-time and safety constraints are met. Indeed, Network Calculus [Chang00,LeBoudec01] has already been used for almost the last 10 years for WCTT evaluation, for instance, to dimension and certify the AFDX network of the A380 (see, for instance, [Grieu04,Frances06]).

1.3 The Pegase project

The French PEGASE project [Pegase10], partially funded by the Agence National de la Recherche (ANR), gathers academics (ENS, INRIA, ONERA) and industrial partners (Thales R&T, Thales Avionics, Thales Aliena Space, RealTime-at-Work) from the aerospace field. It has been undertaken to improve some key aspects of the Network Calculus [Chang00,LeBoudec01] and its implementation, in order to meet increasing requirements in terms of accuracy of the temporal evaluation and size of the systems that are to be studied. Ultimately, the objective is to come up with the techniques and the tools that enable the OEM to dimension an on-board system in the tightest manner (*i.e.* no over-dimensioning) while providing the necessary safety guarantees.
To assess the gains achieved and the practicality of the software tool in an industrial context, 3 case-studies have been undertaken respectively on AFDX [AFDX09], SpaceWire [SpW08] and a NoC. This paper focuses on the use of the PEGASE software tool for the design and validation of the AFDX networks.

1.4 Aims of the PEGASE software tool and its use-cases

The software tool should be versatile in the sense that ultimately it should be usable for research, design as well as certification (by OEMs). These different use-cases imply different requirements from different users. For instance, academics working on the NC theory would want to be able to get access to specific intermediate results that are not needed, and even could be detrimental, when conceiving the networking architecture by design space exploration. On the other hand, proving the correctness of the WCTT results in the certification process might require other kinds of intermediate results to be available.
However, the various functionalities and facets of the software are all based on the solid mathematical foundation that is provided by the NC theory (see section 2). Also, the tool is up-to-date in the sense that, to the best of our knowledge, it implements the latest NC theoretical results. For instance, the tool is able to deal with the so-called ultimately pseudo-periodic (UPP) work arrival functions [Bouillard08], which significantly increases the modeling power and the accuracy of the WCTT results with regard to previous implementations (see Section 4). Being as accurate as possible in the WCTT results is indeed a primary objective of the tool because it has an immediate impact on the architectural complexity and its costs, and thus on the industrial competitively of the developed solutions.
Finally, the tool has been developed with usability in mind. In particular, it includes a graphical user interface (see figures 6 and 7) which allows to set and modify the configurations under study, visualize the results and compare among alternative design choices.

2 Mathematical foundations of WCTT: the Network Calculus

Network Calculus is a mathematical theory designed to compute upper bounds on communication delays and memory usage in networks. It had been successfully used to certify the Airbus A380 AFDX backbone [Grieu04,Frances06]. Here is only a short overview of the method and its benefits in the context of critical embedded systems.

2.1 Network Calculus: history and a recap

The aim is here to provide a general overview of NC, without too many technicalities. The interested reader can refer to [LeBoudec01, Chang00, Boyer10a].
The Network Calculus (NC) theory has been conceived to compute worst case performances of networks, *i.e.* worst delays and worst buffers usage (also known as *backlog*). The first studies [Cruz91a,Cruz91b] were a generalization of common scheduling theories, considering the general notion of *arrival curve* to describe the inputs of a system, instead of the classical "periodic task" model. But the main originality of network calculus, developed from the middle to the end of the 90's, is the parallel made with the filtering theory and the (min,+) algebra [Baccelli92]. Today, network calculus [LeBoudec01,Chang01] is a theory supported by a formal mathematical framework: the (min,+) algebra.
In few words, network calculus handles two kinds of system objects: *flows* and *servers.* A flow is modeled by its cumulative function $R(t)$, which represents the total amount of data produced by the flow from time 0 up to time t. A server S transforms an input flow R into an output flow R', with $R \geq R'$, meaning that the departure time of any bit occurs after its arrival. Such a server does not create any data, neither looses any.
The *backlog* at time t is $b(t)=R(t)-R'(t)$, *that is* the amount of data that has entered into the server and not gone away yet. The *delay* at time t is $d(t)=\inf\{ \tau \mid R'(t) \leq R(t+\tau) \}$ as presented in Figure 1 . The worst backlog and the worst delay can be defined as the maximum backlog and delay for all t. There are formally known as vertical and horizontal deviations, v(R,R') and h(R,R').

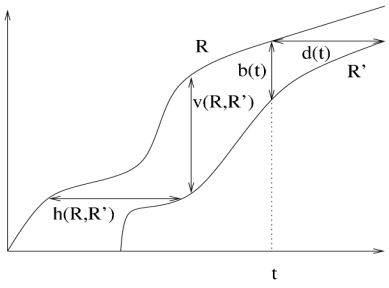


Figure 1: Horizontal and vertical deviation

But the real flows R *and* R' are unknown at design time (they may depend on external events, jitters, varying execution times, etc.). So, network calculus has to compute the bounds based on some safe traffic and service assumptions. In particular it is said that a flow R has an *arrival curve* $\alpha(t)$ if and only if $\forall t,w \geq 0,\ R(t+w)-R(t) \leq \alpha(w)$, meaning that, on any interval of duration w, at most $\alpha(w)$ data are emitted by flow R. The link between (min,+) algebra and network calculus appears when noticing that this condition is equivalent to $R \leq \alpha * R$, where $*$ denotes the convolution in the (min,+) algebra. In a similar manner, a server offers a simple service of curve β iff, for all input flow R, the output flow R' satisfies $R' \geq R * \beta$.

With these hypotheses, the network calculus ensures that the worst delay experienced by the flows R in the server S is bounded by $h(\alpha, \beta)$, while the worst backlog is bounded by $v(\alpha, \beta)$. Moreover, the computation can be propagated, because the theory also gives an arrival curve for the outputs flow, R': $\alpha' = \alpha \div \beta$, where $\div$ denotes the (min,+) deconvolution.
A formal mathematical background also permits to benefit from the general properties of mathematical operators, like commutativity, associativity, monotony, etc. And these properties are of practical interest. For example, if α is an arrival curve for a flow R, any function $\alpha' \geq \alpha$ also is an arrival curve for the flow R, and the symmetric holds for service curve.

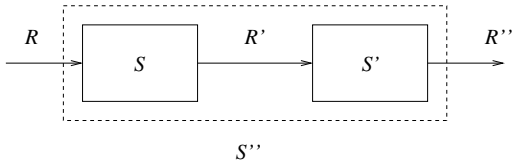


Figure 2: Pay Burst Only Once

A famous result of network calculus, known as "pay burst only once", comes from this mathematical framework: if a flow R goes through a sequence of two servers, S and S' (cf Illustration 2), providing respectively service curves β and β', then, the system is identical to a flow crossing a single server $S;S'$ (where ; is a composition operator) offering a service curve $\beta * \beta'$.
This allows to compute an end-to-end delay, $h(\alpha, \beta * \beta')$ smaller than the sum of the local delays. An interpretation of this result is the following: when considering a bursty flow, at a given instant, the burst can be either in S, either in S', or partially in S and partially in S', but not in both S and S'. Then, the worst delay for S occurs when the burst is in S, the same holds for S', but the global worst case is not the sum of individual worst cases. But going from an intuition to a formal result can be a difficult task. Here, the formal framework, ensuring associativity and commutativity, gives a simple and elegant proof.

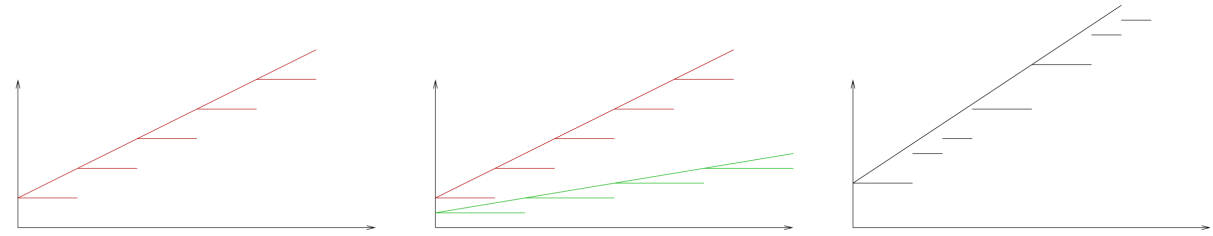
$$R'' \geq R' * \beta' \geq (R * \beta) * \beta' = R * (\beta * \beta')$$

Network calculus also offers results on shared servers, with several policies (First In First Out – FIFO ; Static Priority – SP ; paquetization ; Weighted Fair Queuing – WFQ). Depending on the case, network calculus can compute the exact worst case, or only upper (safe) approximations.

2.2 A formal framework providing flexibility and confidence in the results

One great benefit of the very general framework of network calculus is the built-in ability to model heterogeneous systems, and make safe approximations. It is not related to some periodic task model, as often in real-time systems, with subtle differences between models (offsets, periodic, sporadic, etc). In network calculus, such differences are modeled by different arrivals and/or service curves, which all belong to the same framework. For example, a sporadic flow with packets of size l and inter-arrival T can be modeled by a staircase arrival curve. The sum of two such flows gives a complex periodic arrival curve, with a period equal to the least common multiple of the flows. But network calculus allows to model it in a less precise manner, with an affine curve, as depicted in Illustration 3. Because of the monotony of the underlying mathematical operators, this kind of approximation is safe. Of course, going from staircase functions to linear ones is just an example. Nevertheless the same principle can be applied to non trivial classes of curves. However, there is no free lunch, a simple modeling will require less

computation time but, in general, will also lead to less accurate results. Nevertheless, one can switch from one model to another, depending on the needs, while staying within the same formal framework.



(a) Stair case function and its linear approximation (b) Two functions and their approximation (c) Sum of functions in (b)

Figure 3: Safe approximation, from stair-case to linear function

Network calculus also makes a very clear distinction between the formal world and the real world. As presented in Illustration 4, the principle of formal method is the following: considering a real system Σ, and a property P, the user models the system and creates an object *M* (the model of Σ) in a formal theory, and write an expression Φ (often called property also), such that, if the formal method is able to prove Φ from *M,* then P holds in Σ. The work of the modeler is to build *M* and Φ from Σ and P. The work of the theoretician is to build a formal theory which allows this modeling *and* which provides efficient tools to deduce (or refute) Φ from *M.* Of course, the modeling (arrows 1,3) must be done carefully, but the deduction (2) is robust. In less formal methods, such as some real-time scheduling analyses, there often is some implicit assumptions, some conditions hard to model in the theory, and used in proofs, like "is is clear that the worst case appears when…" or "since such behavior cannot occur…", as illustrated by arrow (5). In network calculus, the formal theory is clear: it is the (min,+) algebra.

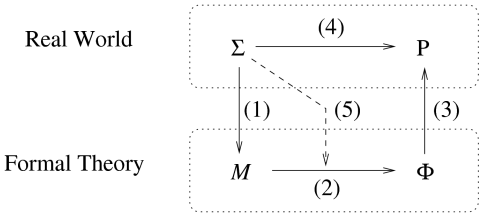


Figure 4: Use of formal theories

3 Description of the PEGASE temporal evaluation tool

The complexity of the targeted systems and of the verification methods imposes the usage of a software tool. But the development of a software tool that implements new mathematical methods and that satisfies the practicability requirements of an industrial context requires preliminary exploratory work and proofs of feasibility. Furthermore, researchers need tools that allow them to evaluate the relevance of the theoretical findings through concrete computations on industrial case-studies. For this reason, a prototype is being developed during the project and it is expected to facilitate a rapid transfer of the outcomes of the project to the industry.

3.1 Requirements on the tool

The practicability of such a tool in an industrial context depends on several aspects:

- acceptance by the certification authorities,
- contained computation time to obtain the results,
- domain-specific support for creating system descriptions that helps to avoid modeling errors,
- ease of understanding and visualization of the analysis and optimization results.

The usefulness of such a tool in an academic context depends on two main aspects:

- models that are as general as possible - even if it is to the detriment of raw performance,
- extendibility that enables exploratory work.

Because of such sometimes conflicting requirements, the tool has been designed in a modular way, as presented in next subsection. In a certain extent, the PEGASE tool can be seen as a modular framework, with different sub-tools, having different sub-goals, linked together by a common theoretical and implementation framework, as presented in the next paragraph.

3.2 Architecture of the tool

Different users will give different importance to different requirements. Thus, the design allows to have different implementations for different users.

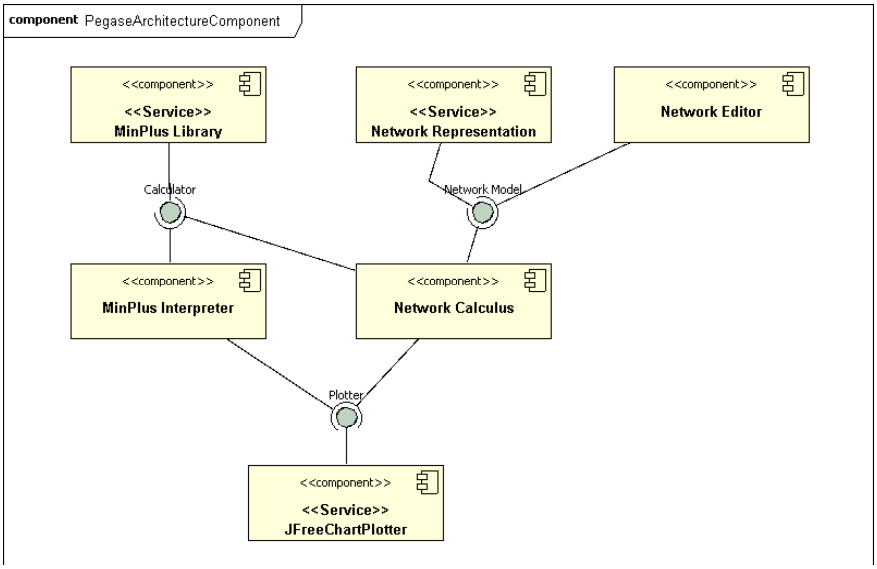


Figure 5: Components of the prototype (UML notations)

As an example, let us consider the *MinPlus Library* component. Network Calculus uses (min,+)-algebra operations whose complexity is strongly dependent on the considered class of arrival and service curves. The more specific the class of curves are (sufficient for industrial applications or coarse-grained results), the lower the complexity[2], while the more general the class of curves (as needed for research), the higher the complexity. For example, the first works on NC [Cruz91a] were only using affine curves (like shown in Figure 3), where * and ÷ can be computed in constant time and less than ten lines of code. A very general class of piece-wise affine functions, called Ultimately Pseudo Periodic or UPP for short, has been studied in [Bouillard08]. These UPP functions require complex and computationally intensive algorithms to work with, while the simple concave piece-wise affine functions (that belong to the ICC class, for Increasing Convex or Concave) involve only linear algorithms (see [Boyer08]).
In PEGASE, both ICC and UPP have been developed; using either floating points or exact rational numbers, and the user is free to choose the *MinPlus Library* component that best suits its needs. Table1 gives, for each module, the lines of code (LOC), the complexity (measured by the cyclomatic complexity), the number of methods and the ratio complexity/methods.

Module	#Lines of code	Complexity (Cyclomatic)	#Methods	Cplx/#Methods
Fraction	816	268	73	3.67
Double	84	32	24	1.33
ICC	1292	318	74	4.3
UPP	3416	719	106	6.8

Table 1: Size and complexity of the implementation. The code shared by all modules is not indicated here: it represents 2062 lines of code, 101 methods for a complexity equal to 504. Cyclomatic complexity is a software metric that indicates the complexity of a program by measuring the number of paths in the source code.

At the time of writing, all modules are operational and their core features have been validated. The features needed for the most advanced use-cases are currently conceived and implemented.

[2] Complexity should be understood here with two meanings: computational cost *and* implementation cost. A simple class of functions can have low computational cost (linear) and be a good choice in a coarse design phase, while, at a later stage of design, a more general class leading to more precise results can be a better choice.

For instance, we are developing a design space exploration module that will help the OEM to optimize the design choices regarding the topology, routing and allocation of the functions.

3.3 The Network Editor

The code of the GUI is mainly not hand-written but generated from a high-level specification in UML with RTaW-Generator (see [RTaW08]). This was especially useful at the beginning of the development process, when the interactions with the users lead to frequent modifications. Another interest is that the code quality is ensured by the generator, which has been validated on several large projects.

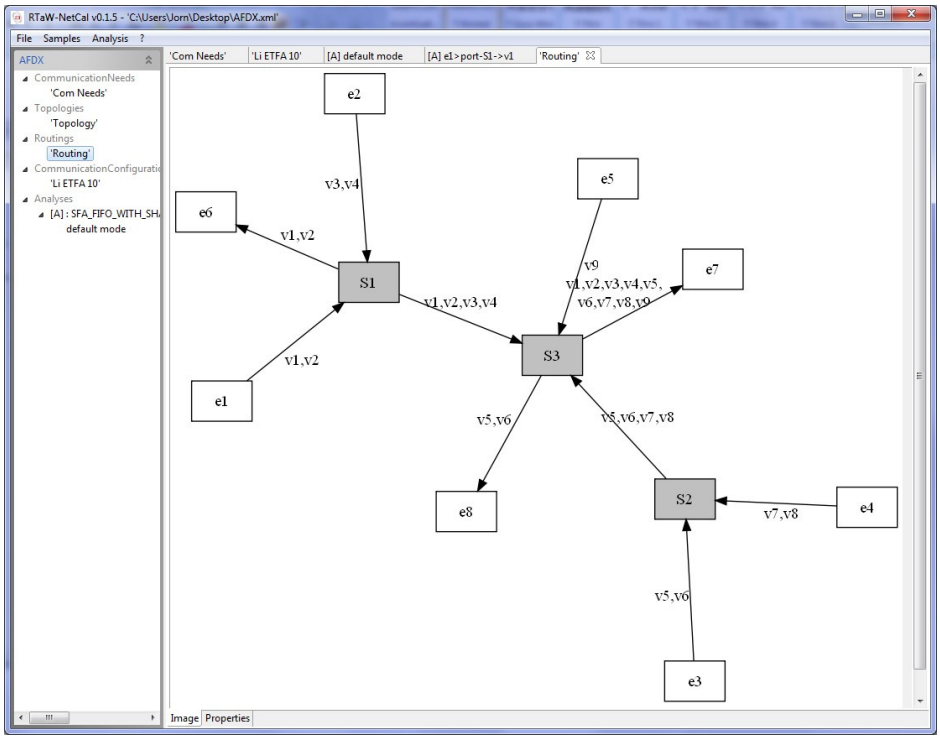


Figure 6: Topology of the AFDX network. The gray boxes are the switches while the end systems are the white boxes. The names of the virtual links are shown as labels of the physical links.

The GUI gives the user access to the whole system description, in order to visualize or set parameter values concerning the topology of the networks, the characteristics of the switches and virtual links, etc. As explained in §3.2, not all users will need to run the computations in the same manner, therefore the desired tradeoff between speed and accuracy can be chosen through the GUI. The tool can not only be used in an interactive manner, computations can be run through scripts.

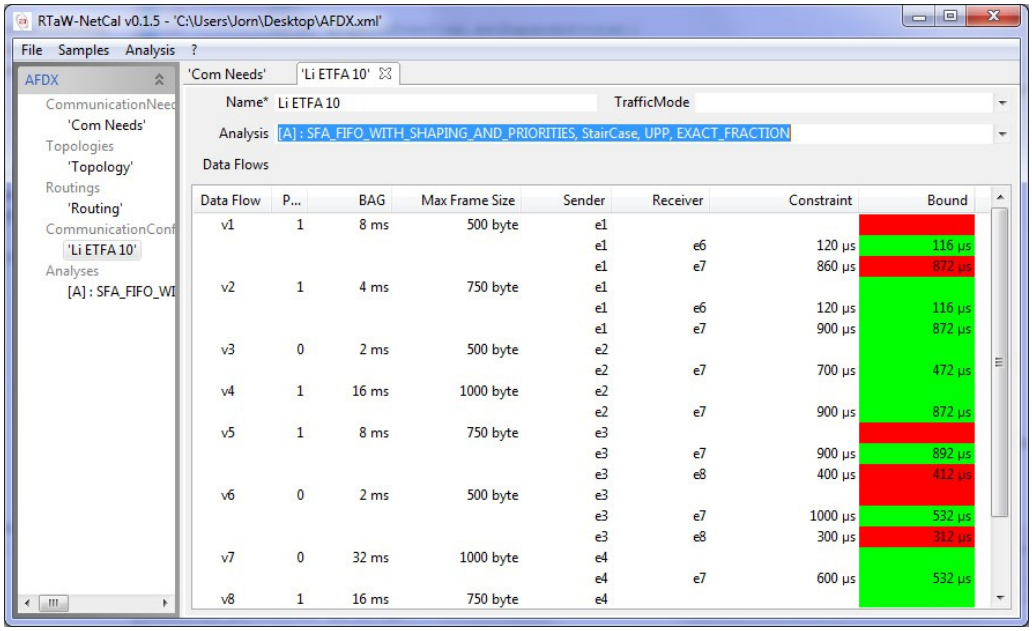


Figure 7: results panel showing the computed Worst-Case Traversal Times, where red means that the time constraint cannot be guaranteed for a given virtual link.

3.4 Tool validation

Java has been chosen as programming language for its lower risk for programming errors Furthermore continuous integration with frequent releases is performed in order to get rapid feedback from the academic and industrial partners. Given the safety requirements of the application domain, a particular effort is put on the validation of the code:

- Numerous unit tests of the different components of the tools with the mandatory objective of 100% of source code coverage,
- Static analysis of the source code with the tool SONAR, with the objective to remove all identified warnings,
- Extensive automated comparison tests with the Network Calculus tool NC-maude [Boyer10b].

4 Performance evaluation

4.1 System setup

The performances of the algorithms and their implementation in the tool have been evaluated on an industrial size example provided by Thales Avionics. The following table summarizes the main characteristics of the modeled communication system.

Entities	Number
End Systems	104
Routers	8
Virtual Links	974

Latency constraints	6501

As can be seen in the following table, each Virtual Link (VL) has on average 6 destination end systems. This explains the 6501 latency constraints shown in the first table, which means also that 6501 WCTT bounds need to be computed.

	Virtual Link destinations	**BAG (minimum interarrival time)**	**Maximal Packet Size**	**Traversed Routers**	**Latency Constraints**
minimum	1.0	2 ms	100 bytes	1	1000 µs
average	6.6	60 ms	380 bytes	1.3	10040 µs
maximum	84.0	128 ms	1500 bytes	4	30000 µs

4.2 WCTT evaluation algorithms and their running times

Bounds on WCTT have been computed with two different kinds of initial arrival curves, with two different numerical types and with two different function types. The following table summarizes the advantages and disadvantages of these analysis options.

		Advantages	**Disadvantages**
Number type	**Floating point**	Faster execution of min-plus operations.	Rounding errors and incompatibility with UPP function class.
	Fraction	No rounding errors and compatibility with all function classes.	Slower execution of min-plus operations.
Function class	**ICC**	Implementation of min-plus operations are less complex and thus their execution is faster	Tighter stair case arrival functions cannot be represented and thus bounds on WCTTs are larger.
	UPP	Tighter stair case arrival functions can be represented and thus bounds on WCTTs are tighter.	Implementation of min-plus operation much more complex and thus their execution is slower.
Initial arrival function kind	**Token bucket**	Simple structure which is compatible with all function classes.	Looser arrival function which leads to larger bounds on WCTT.
	Stair case	Complex structure that is not compatible with all function classes. Can only be handled with UPP functions.	Tighter arrival function which leads to tighter bounds on WCTT.

The following table shows the duration of the WCTT calculations for all possible configurations combinations[3].

[3] Experimentations have been run on a 2.5Ghz Intel processor running under a 64 bits- Linux, with the Sun/Oracle 1.6.0_24 java machine.

Configuration number	Initial arrival function	Number type	Function class	Duration of WCTT computations
1	Token bucket	Double	ICC	2 s
2	Token bucket	Fraction	ICC	11 s
3	Token bucket	Fraction	UPP	19 s
4	Stair case	Fraction	UPP	33 mn

From the computation time point of view, as expected, the calculations are much faster:

- if based on floating points instead of fractions,
- if based on the ICC function class instead of the UPP function class,
- if based on token-bucket arrival-functions instead of stair-case arrival functions.

4.3 Results accuracy

The tightness of the calculated bounds behaves in principle exactly in the opposite of the running time, but differences in accuracy might be in some cases negligible or null. For instance, changing only the numbers representation from fraction to double, while keeping the ICC function class and the token-bucket arrival function, lead to negligible differences (below µs level) in the case of the studied real-world sample system (configurations 1 and 2). As expected, changing only the function class (ICC or UPP), while keeping the initial arrival function and the number representation identical leads to identical results (configurations 2 and 3).

However, changing the initial arrival function to stair-case, produces significantly tighter bounds, with improvements ranging from 0% up to 20%, and an average gain equal to 6%. As can be seen in Figure 8 where the computed WCTT bounds of the virtual links are sorted in increasing order their values, improvements are basically proportional to the bounds.

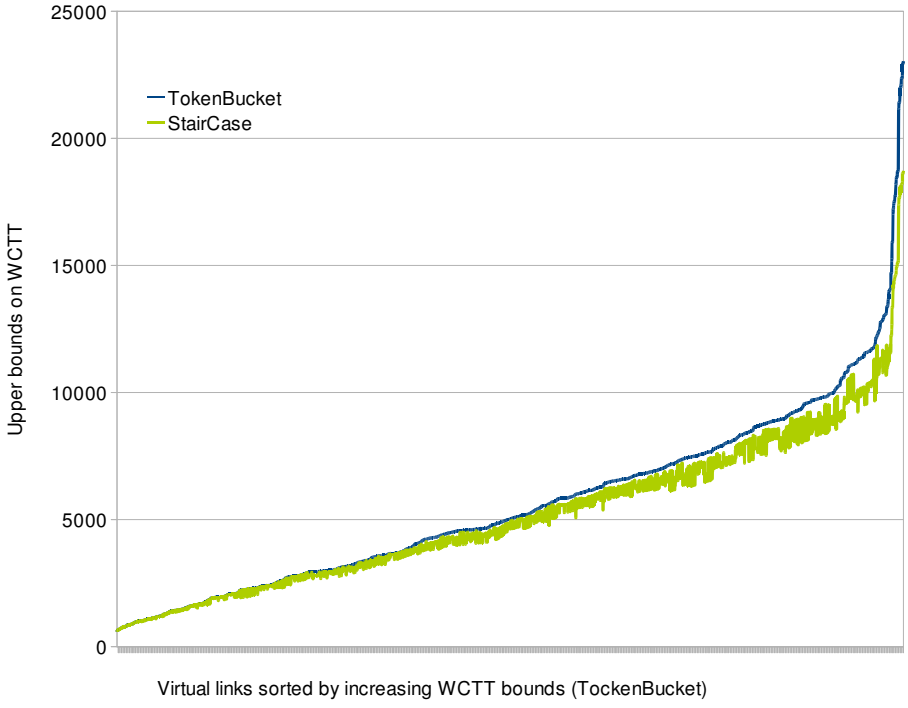


Figure 8: Worst-Case network Traversal Times obtained with Tocken Bucket and Stair Case modeling of the input traffic.

The priority of the VLs is one the factors that influences the gain achieved with the stair case input traffic model. On Figures 6 and 7, one observes that the lower the priority, the higher the improvement brought by stair-case: 0.38% of average improvement for the highest priority VLs versus 12.5% for the lowest priority VLs.

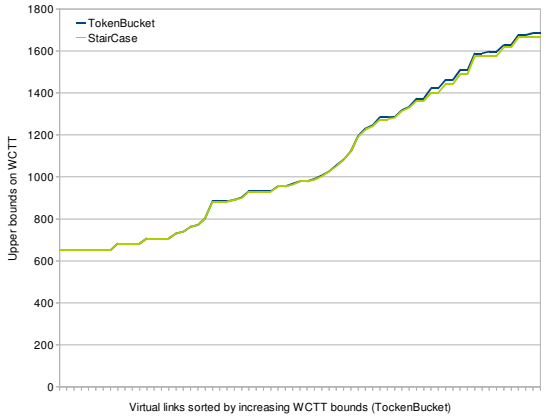


Figure 9: WCTT of higher priority VLs

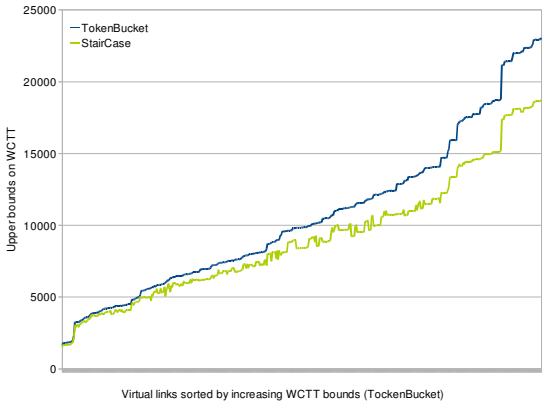


Figure 10: WCTT of lower priority VLs

The same analysis can be done for the length of the VL path (*i.e.* the number of switches between sender and receiver). As can be seen on Figures 8 and 9, the conclusion is that the longer the path, the higher the improvement: 5.7% of average improvement for paths of length 1 versus 7.3% for paths of length 3.

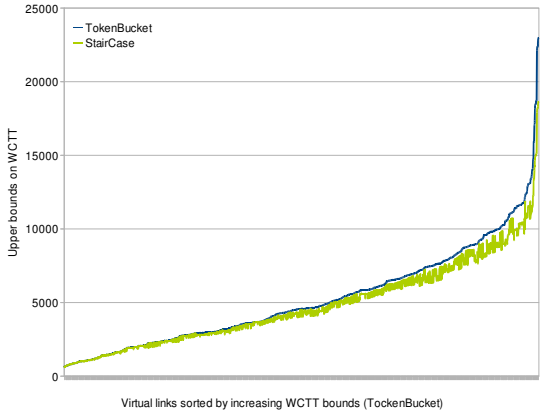


Figure 11: WCTT of VLs crossing 1 switch

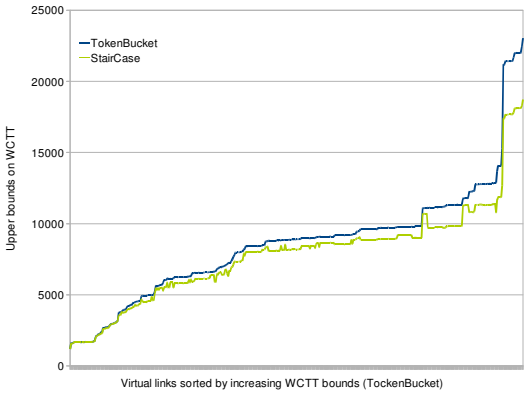


Figure 12: WCTT of VLs crossing 3 switches

5 Conclusions and future work

For the last 10 years, Network Calculus has proven to be a powerful formalism that is well suited to provide guarantees on the worst-case performances of large critical embedded systems, such as airplanes. Thanks to recent theoretical and algorithmic improvements, such as the ones that are being obtained in the Pegase project, it becomes possible to achieve significant gains in accuracy, reducing thus the over-provisioning of resources, and provide better support for design space exploration techniques.

One of the main objectives of the Pegase project is to come up with a practical and efficient software tool suited for both research and industrial use at different steps of the design. The features that allow the basic use-cases of the tool have been implemented and successfully tested on realistic case-studies. Our ongoing work is to implement more complex use-cases, related to design optimization, and benchmark our algorithms against other techniques such as the trajectory based approach [Martin04]. Also, we believe there are other classes of functions that may have the necessary power for modeling avionics systems without the complexity of the most general UPP class.

6 References

[AFDX09] AEEC, "Arinc 664p7-1aircraft data network, part 7, avionics full-duplex switched Ethernet network". Airlines Electronic Engineering Committee, September 2009.

[Baccelli92] Francois Baccelli, Guy Cohen, Geert Jan Olsder, and Jean-Pierre Quadrat, "Synchronization and Linearity: An Algebra for Discrete Event Systems", volume ISBN; 978-0471936091. John Wiley and Son, 1992. http://cermics.enpc.fr/ cohen-g//SED/book-online.html.

[Bouillard08], Anne Bouillard, Éric Thierry, "An Algorithmic Toolbox for Network Calculus", Journal of Discrete Event Dynamic Systems, Vol. 18(1), pages 3-49, 2008

[Boyer08] Marc Boyer and Christian Fraboul. Tightening end to end delay upper bound for AFDX network with rate latency FCFS servers using network calculus. In Proc. of the 7th IEEE Int. Workshop on Factory Communication Systems Communication in Automation (WFCS 2008), pages 11-20. IEEE industrial Electrony Society, May 21-23 2008.

[Boyer10a] Marc Boyer. Half-modeling of shaping in FIFO net with network calculus. In Proc. of the 18th International Conference on Real-Time and Network Systems (RTNS 2010) , Toulouse, France, November 4-5 2010.

[Boyer10b] Marc Boyer. NC-maude: a rewriting tool to play with network calculus. In T. Margaria and B. Stepen, editors, Proceedings of the 4th International Symposium On Leveraging Applications of Formal Methods, Verication and Validation (ISoLA 2010) , LNCS. Springer, 2010.

[Chang00] Cheng-Shang Chang. Performance Guarantees in communication networks. Telecommunication Networks and Computer Systems. Springer, 2000.

[Cruz91a] Rene L. Cruz. A calculus for network delay, part I: Network elements in isolation. IEEE Transactions on information theory ,37(1):114-131, January 1991.

[Cruz91b] Rene L. Cruz. A calculus for network delay, part II: Network analysis. IEEE Transactions on information theory , 37(1):132-141, January 1991.

[Frances06] Fabrice Frances, Christian Fraboul, and Jerôme Grieu. Using network calculus to optimize AFDX network. In Proceeding of the 3thd European congress on Embedded Real Time Software (ERTS06), Toulouse, January 2006.

[Grieu04] Jerôme Grieu. Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques . PhD thesis, Institut National Polytechnique de Toulouse (INPT), Toulouse, Juin 2004.
[LeBoudec01] Jean-Yves Le Boudec and Patrick Thiran. Network Calculus , volume 2050 of LNCS, Springer Verlag, 2001.
[Martin04] Steven Martin, Pascale Minet, Laurent George, "The trajectory approach for the end-to-end response times with non-preemptive FP/EDF*", Proceedings of the Int. Conf. on Software Engineering Research and Applications (SERA'04). Volume 3647 of LNCS., Springer, 2004.
[Pegase10] M. Boyer, PEGASE project home page. http://sites.onera.fr/pegase, 2010.
[RTaW10] RealTime-at-Work, "Minplus-Console: a (Min,+) algebra interpreter for Network Calculus", available for downloadable at url: http://www.realtimeatwork.com/downloads/.
[RTaW08] RealTime-at-Work, "RTaW-Generator: code and GUI generation from UML specifications", see: http://www.realtimeatwork.com/software/rtaw-generator/, 2008.
[SpW08] ECSS, "Spacewire – links, nodes, routers and networks", ECSS-E-ST-50-12C, European cooperation for space standardization (ECSS), ESA-ESTEC, Requirements & standards division, July 2008.