The PEGASE project: precise and scalable temporal analysis for aerospace communication systems with Network Calculus*

Marc Boyer[1], Nicolas Navet[2], Xavier Olive[3], and Eric Thierry[4]

[1] ONERA, 2 av. E. Belin 31055 Tououse Cedex 4, France
Marc.Boyer@onera.fr
[2] INRIA / RealTime-at-Work (RTaW), 615 rue du Jardin Botanique, 54600 Villers-lès-Nancy, France
Nicolas.Navet@realtimeatwork.com
[3] Thales Alenia Space, Research Department, 100 bd du Midi, 06156 Cannes La Bocca Cedex xavier.olive@thalesaleniaspace.com
[4] LIP, ENS Lyon, 46 allée d'Italie, 69364 Lyon Cedex 07, France
Eric.Thierry@ens-lyon.fr

Abstract. With the increase of critical data exchanges in embedded real-time systems, the computation of tight upper bounds on network traversal times is becoming a crucial industrial need especially in safety critical systems. To address this need, the French project PEGASE grouping academics and industrial partners from the aerospace field has been undertaken to improve some key aspects of the Network Calculus and its implementation.

1 Introduction

Critical real-time embedded systems (cars, aircrafts, spacecrafts) are nowadays made up of multiple computers 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, the time between sending and receiving a message depends not only on the technological constraints, but mainly on the interactions between the different streams of data sharing the media.

It is therefore necessary to have techniques to guarantee, in addition to local scheduling constraints, the worst case traversal time of the network (WCTT) and thus ensure a correct global real-time behaviour of the distributed applications/functions. If the temporal evaluation techniques used are too pessimistic, it leads to an over-dimensioning of the network which involves extra cost, weight and power consumption. In addition to being precise, these verification techniques must be scalable. For instance, in a modern aircraft, thousands of data

* This work has been partialy funded by French ANR agency under project id ANR-09-SEGI-009.

streams share the network backbone and therefore the complexities of the algorithm should be at most polynomial.

In the French PEGASE project, we aim to improve the theory of Network Calculus [1, 2], which has already been used to certify the AFDX network of the A380, and its algorithmic implementation in order to meet the scalability and tigthness requirements. To assess the gains achieved and the practicability of the software tool in an industrial context, 3 case-studies have been undertaken respectively on AFDX [3], SpaceWire [4] and a NoC.

Section 2 presents the industrial context of embedded real-time networks. Section 3 provides a recap of the main techniques that can be used to compute worst-case traversal times. The case-studies of the project are described in section 4. The last two sections of the paper gives an overview of the first results of the project: some theoretical results in section 5, and the current version of the tool in section 6.

2 Industrial Context

2.1 Wide-scale communicating systems

As described in the introduction, modern real-time critical embedded systems are composed of dozens to hundreds of electronic equipments (including computers, smart sensors...), communicating through thousands of data flows. To guarantee the correctness of the system's functions, real-time behavior of each application on each equipment must be ensured (with schedulability analysis), but the temporal validity of the data consumed (their "freshness") must also be managed, that is to say, the delivery of the data must be guaranteed and the delay introduced by the network must be bounded.

2.2 Shared resources: homogeneous vs heterogeneous flows

To reduce weight and integration costs, networks are shared resources. But this sharing could have different degrees. In a first industrial step, networks are shared by homogeneous flows: one network for control command flows (small data, small bandwidth, need of low latency), and another for mission data flows (large images in spacecraft without real time constraints but guarantee of delivery is needed). In a second step, the network can be shared by heterogeneous flows (control command *and* mission data). One can also distinguish critical and non critical flows (control command versus entertainment in aircraft).

Mixing critical and non-critical, control/command and mission-specific streams may lead to significant gains. It is the opportunity to have a single on-board data network providing thus some significant savings. Besides, the margins can be factorized, reducing de-facto the need of over-sizing the system to fulfill the dependability requirements. Some other elements speak also in favor of this heterogeneous data mix, and all permits to save some time and money in general. Nevertheless, the validation or certification of such network architecture requires providing for each flow a bound on data delivery, and the proof of segregation (i.e. no interaction) between the flows.

2.3 Mono-segment vs multi-hop (homogeneous or heterogeneous)

The last point is the evolution from mono-segment to multi-hop networks. If the system is small enough, all the nodes can be connected to the same segment. There are numerous network technologies suited for this type of architecture, for example, MIL-STD-1553, CAN, TTP, FlexRay, etc, but each one has limits on the global bandwidth and the number of equipments that can be connected[5]. Therefore, a complete system could be too large for a single segment. In this case, a multi segments network must be built, interconnecting segments with gateways or switches. In this context, an end-to-end communication can be multi-hops.

Moreover, multi-hop communications can be homogeneous (with the same technologies at each segment, like the AFDX or SpaceWire technologies), or heterogeneous: a sensor could be connected to a low-bandwidth segment, interconnected to a system backbone, where nodes reading the sensor values are connected. The real-time communication is, in this case, multi-hop and heterogeneous.

2.4 Use of formal methods in the development process

If computing WCTT is an important issue in designing critical real-time systems, this is not the only one from a global industrial point of view: three others (at least) should be considered: design complexity, method stability and method simplicity.

Verification of a platform is the second half of a problem: building a configuration is the first half. From an industrial point of view, a system too hard to be (efficiently) configured can be rejected, even if there are methods to verify it. For example, priority levels are a way to ensure performances and segregation, but they are not an intrinsinc characteristic of an applicative data flow. It might be difficult to map, in an efficient manner, thousands of data-flows onto dozens or hundreds of priority levels.

There is also a trade-off between complexity of the verification and confidence: if a formal method or a model is too complex, it is prone to contain errors, or its implementation tool might have bugs. From the industrial point of view, the simplest method to guaranty/certify the WCTT will be the easiest to adopt.

3 Related works

The need for formal models to compute end-to-end bounds on delays and buffer consumptions is relatively new in embedded systems, due to a change in the technologies used. Until recently, the embedded bus technologies were quite simple,

[5] Sometimes, such as for TTP or FlexRay, is is required to have a distributed global clock over all components, which is usually a strong constraint in large-scale systems.

leading to constant or at least easy to bound delays (ARINC429, MIL-STD-1553, CAN). But the introduction of new technologies (AFDX, SpaceWire), more efficient in some ways but also more complex, creates the need for new and/or improved analysis methods. These models must be correct, i.e. the computed bounds must be true bounds (possibly over-estimated). Moreover, they should be as tight as possible, i.e. not too pessimistic (otherwise, this leads to an over-dimensioning, extra weight and power consumption). Finally, they should be efficient and scalable, to handle modern embedded architectures with hundreds of computers and thousands of flows.

3.1 Main approaches to timing verification

The timing verification approaches can be classified into three main categories: methods from real-time systems (often industrial ones), methods from computer networks performances (like Internet, ATM...) and methods from timed systems (model-checking mainly).

- Model-checking is correct, tight, but not scalable. There exist very-efficient timed model-checkers [5], and even if they are not designed to compute delay, the delay can be found by trying to verify a property like "The message arrives before date D", and to compute the bound by a dichotomic search on D. Such model-checkers have been extended to perform parametric verifications [6] which allows to compute D directly. But even if great advances have been done in algorithms and tools efficiency, model-checking still suffers from the combinatorial explosion of the state space, and cannot be used for large systems.
- Real-time scheduling is a wide research area, with a long history and a lot of diverse results. If most results in real-time scheduling have been used for local scheduling, or distributed scheduling with the network delays as an input of the problem, some studies have been done to handle communications [7], (which associates some well known local methods with a fixed point iteration) or to see one bus as a local scheduling problem [8]. More recently, approaches known as "trajectorial" have been developed [9, 10]. Such approaches give good results [10], but the complexity is relatively high: computing the release time of a message created at time t requires solving a non-linear system, and this computation must be repeated for each significant instant, whose number depends on the least common multiple of the period of the flows.
- Network Calculus is a theory to get deterministic upper bounds in networks that has been developed by R. Cruz [11, 12], and popularized with two books [1, 2]. A nice overview of can be found in [13]. It is mathematically based on the (min,+) dioid and from the modelling point of view, it is an algebra for computing and propagating constraints given in terms of envelops. A flow is represented by its cumulative function $R(t)$ (that is, the amount of data sent by the flow up to time t). A constraint on a flow is expressed by an arrival curve $\alpha(t)$ that gives an upper bound for the amount of data that can be sent during any interval of length t. Flows cross service elements that

offer guarantees on the service. A constraint on a service is a service curve $\beta(t)$ that is used to compute the amount of data that can be served during an interval of length t. It is also possible to define in the same way minimal arrival curves and maximum service curves. Then such constraints envelop the processes and the services.

3.2 Why Network Calculus fits embedded systems

Among the other temporal verification techniques, Network Calculus fits well critical embedded systems for several reasons. First of all, it relies on strong mathematical foundations since network calculus is based on the (min,plus) algebra [14] with well-identified mathematical assumptions.

Most often, classical scheduling-based methods are based on the exhibition of a “worst case scenario”, which must be translated into an analytical expression, that in turn should be solved to obtain numerical results. This process mainly relies on human reasoning, which could lead to errors, as it happened for the CAN schedulability analysis, considered as solved in [15], and which was found to be slightly flawed and corrected 13 years later in [8]. With its strong mathematical background, Network Calculus is less sensitive to this kind of problem but it is of course sensitive to mathematical aspects, like continuity, limits, etc.

Network Calculus also handles natively multi-hops networks, one of its famous result is the “pay burst only once”, which allows seeing multiple hops as a single element. Since this is a generic method, it also handles heterogeneous networks and allows to consider an end-to-end path with, for example, a CAN and an AFDX segment.

A last property is of special interest: Network Calculus allows a lot of sound over-approximations[6]. Such feature enables to reduce the computation time (at the expense of the results accuracy), which could be useful for a coarse-grained design or to evaluate large-scale systems. But, on top of it, it lets the user opt for a simple model, or simple verification algorithms, if the more accurate and complex ones are too hard to verify, qualify or certify (depending on the industrial context).

Of course, these benefits are counterbalanced by the pessimistic approximations made by the theory.

3.3 Network Calculus : an overview of the state of the art

In its simplest form, Network Calculus enables to perform the following operations:

– compute the exact output cumulative function or, at least, bounding functions,
– compute output constraints for a flow (like an output arrival curve),

[6] Soundness, in this context, means that the bound computed in the over-approximated model are still bounds, probably pessimistic but always valid.

- compute the remaining service curve, that is, the service that is not used by the flows crossing a server,
- compose several servers in tandem,
- give upper bounds on the worst-case delay and backlog (bounds are tight for a single server or a single flow).
- the operations used for this are an adaptation of filtering theory to (min,+): (min,+) convolution and deconvolution, sub-additive closure.

These possibilities have been extended in several directions. First the notion of service curve can be defined in several manners, depending on the chosen model. To compute remaining service curves, one has to work with strict service curve [16], but to study FIFO scheduling policy, the notion of service curve is enough [17]. Real-Time calculus is mainly based on Network Calculus, but also uses concepts of real-time scheduling theory [18]. For this approach, the elementary operator transforms an arrival curve of a flow and a service curve of a service elements into the output arrival curve and a remaining service curve, and this operator has good compositional properties to study systems with fixed priorities. Finally, there are some adaptations of Network Calculus to Stochastic Network Calculus [2, 19]) in order to relax a bit the constraints, but worst-case delay bounds cannot be computed with this theory.

Concerning the results achieved in the Network Calculus field, a lot of studies are now focused on computing performances in networks and composing network elements. Up to our knowledge, the networks that have been extensively studied are: servers in tandem or sink trees [17, 20, 16]. Some other studies focus on general networks with cyclic dependencies, and exhibit networks where, against intuition, the load is very small in each server but can be unstable (that is, the backlog is not bounded). To get such results, the authors construct ad-hoc scenarios [21]. More compositional methods have been used very recently for both Network Calculus and Real-Time Calculus, giving sufficient conditions to get stability and worst-case delay upper bounds [22, 23].

The prominent software tool is Rockwell-Collins ConfGen Tool which is a proprietary tool that uses network calculus to compute traversal times on AFDX. It has been for instance used to validate the delays in AFDX network for Airbus A380. The bounds on end-to-end delays provided by this tool were really larger than what was observed on simulations and experimentations [24]. The PEGASE proposal aims to develop an ambitious successor of this tool providing several improvements and additional features: more realistic bounds, handling of cyclic dependencies, new class of protocols (e.g., wormhole routing) and design assistance. These features must be based on new theoretical investigations.

3.4 Objectives and novelty of the PEGASE project

The main idea of the family of Network Calculus techniques is to take into account the traffic characteristics under the form of regulating functions and to model network nodes (switches, filters, cables) as operators acting on those functions.

Thus, Network Calculus operates at the flow level and not at the packet level (unlike the trajectorial approach, for example) and never uses the state space of the system (unlike model checking). These two features of Network Calculus make it perfectly fit to analyse systems whose state spaces are very large or with a combinatorial complexity making them hard to analyse with approaches based on a fine description of their behaviour.

This project has two objectives, one is rather theoretical and aims at improving Network Calculus in terms of control of bounding errors and assertion of its descriptive power. The other is to demonstrate its usefulness for the design of communicating embedded systems, especially for the aeronautic and the space industry.

Industrial objectives: simplify and tighten. This project focuses on the communication networks of embedded real-time systems, where the worst end-to-end delay must be characterised and taken into account. The industrial objectives are:

- tighten the computed worst end-to-end delay to reduce over-provisioning which leads to extra weight and power consumption,
- simplify the design of such networks (dimensioning, routing) to reduce conception costs,
- provide some analysis tools to facilitate the validation of critical real time embedded data networks transporting different classes of traffic (guaranteed delivery in time, assured delivery, best effort. . .).

Scientific objectives: pushing the limits of Network Calculus. To achieve these industrial goals, we propose to work on three scientific axes:

1. Eliminate some current over-approximations to tighten the bounds: some preliminary works have shown that the difference between the computed bounds and the real worst case comes from the poor modelling of some characteristics of embedded network elements. For example, the non-preemptive aspects are handled in a pessimistic way. Furthermore, the global analysis is often done by the recursive use of local results, and it has been shown in [16] that it could produce very pessimistic results. New global methods should be developed.
2. Extend the class of systems that can be modelled and analysed: up to now, some systems can not or incompletely be handled with Network Calculus (cyclic dependencies between flows, wormhole routing). So far, such configurations must be avoided if one wants to apply Network Calculus formulas. We aim at removing these restrictions.
3. Provide help in the design (dimensioning, routing): in the current network design process, the worst end-to-end delay is taken into account a posteriori (once the system is designed, worst-case bounds can be computed to validate the design). On the opposite, in classical scheduling theory, there

exist heuristics to help the design (priorities, placement). We would like to develop similar methods in the Network Calculus framework to help with dimensioning and routing in embedded networks.

There are of course scientific hard points, the main challenges addressed by PEGASE can be grouped into three main points:

1. Semantics relations: Network Calculus have been extended in various directions for various purposes, but the differences between model hypotheses are not always clear. In particular, they all use notions of arrival curve and service curve, sometimes without precise definitions of the assumptions or without discussions about the conditions of application. What real system behaviour can be modelled by each definition? Some preliminary results can be found in Section 5.
2. Modelling granularity: one main asset of Network Calculus is the liberty in the modelling granularity. A complex behaviour can always be approximated by a lower (resp. upper) service curve (resp. arrival) curve. Such approximation is conservative, i.e. the results obtained with the coarse model are still valid but often not very tight. But, in general, the precise modelling is too computation consuming. Then, a trade-off between computation cost and tight modelling must be find.
3. Are shared memory flow control protocols (min,plus) systems?: All Network Calculus flavours are based on some (min,plus) theory. But it is not clear whether flow control protocols with shared memory (like wormhole routing used in SpaceWire or some NoC) are such kind of systems.

4 Case-studies : AFDX, SpaceWire and NoC

To assess the extent to which network calculus can be an effective tool in various industrial contexts, three real applications based on distinct communication protocols are used as case-studies during the project.

4.1 AFDX

The AFDX technology (ARINC 664 standard, part 7, [3]) is an embedded network based on the Ethernet technology, in order to take benefit of a largely deployed technology and reach acceptable costs. Ethernet offers large bandwidth, but suffers from indeterminism (in particular, the well known random back-off algorithm involved in collision). In AFDX, each end-system is connected to a switch with full-duplex links: there is no more collision, and indeterminism comes only from waiting time in switch shared queues. To get guarantees on this shared network, incoming flows must respect some traffic contract. In AFDX, this contract is a *Virtual Link* (VL). Each VL is a static multicast mono-source route in the net (a tree), with a priority level, and three parameters related to bandwidth: a minimal frame size, a maximal frame size, and a *Bandwidth Allocation Gap* (BAG) which is the minimum time between two frame emissions by the source.

A typical AFDX network could have a dozen of switches, hundred of connected equipments, thousand of VL, and a total incoming throughput between 150Mb/s and 200Mb/s. Given an AFDX configuration, it is possible to compute upper bounds on delays and buffer sizes (see for instance [25, 26]), however there are still some challenges left:

– more accurate temporal verification that enables to obtain the same guarantees on performance with less embedded switches,
– automatic configuration design.

4.2 SpaceWire

The purpose of the SpaceWire standard [4] is:

– to facilitate the construction of high-performance onboard data-handling systems,
– to help reduce system integration costs,
– to promote compatibility between data-handling equipment and sub-systems,
– to encourage re-use of data handling equipment across several different missions.

The SpaceWire network enhances the communication capabilities, but induces more complexity in the traffic management, so requiring enhancement of classical methods of validation. In addition, high speed network (for command/control data) is a new field of embedded designs for Space. Traffic complexity increases is due to (i) the presence of routers and switch matrices (presence of multiple application data and sources, wormhole routing); (ii) the overheads introduced by multiple layered protocols; (iii) the SpaceWire standard features (various and high data rates, no bus controller, arbitration within routers, non predictable dispersion in delays, links shared by different data flows...). The new challenges for SpaceWire adoption are about:

– Defining adequate network topology (no bottleneck, redundancy),
– Consolidating communication designs and performances (latency delays),
– Sizing local resources (temporary storage).

Thales Alenia Space provides, as case study, a long-term horizon architecture based on a single and common SpaceWire network for the transport of command/control and mission (science, observation) data. It consists of 14 nodes (9 instruments and 5 platform equipments) and 4 routers. The Figure 1 shows an overview of the architecture. It deals with 39 data flows (9 mission flows and 30 control/command flows). Flows are asynchronous and follows a known policy for production / consumption of data.

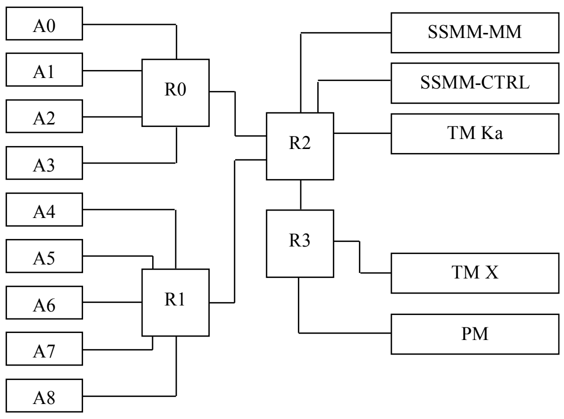

Fig. 1. SpaceWire architecture.

4.3 Network on Chip

Integration is a multi-level concern in embedded systems. Shared network is now the current solution on embedded chips, either as interconnecting different devices on a single chip (system on chip - SoC) or replacing the bus on multi-core CPU. To compute the worst case execution time (WCET) of code on such hardware, the delay introduced by this internal communication medium must be taken into account. But current trend in chips is to increase performances and reduce predictability. In critical system, it makes no sense to have a chip with high mean performances if the worst case cannot be reasonably bounded. This case study is the most prospective one of the project: several hard issues have been identified. To begin, there are a lot of routing technologies in NoC, and some could be easier to verify than other with network calculus. A first issue is to choose and/or define a manageable NoC. Second main issue is the characterization of the data flow exchanged by the different components in order to help the mapping of the application (Design Space Exploration)

5 Some theoretical improvements

The first months of the project have been spent to clarify the relationships between the main variants of envelope-based models, like the Real-Time Calculus [18], variable capacity nodes, strict service and minimum service [27]. Some restrictions on strict priority (SP) residual service have also been lifted. Tight bounds have been obtained for the FIFO policy [28, 29] with results on the complexity of computing these bounds [29].

5.1 Model hierarchy

Considering that it exists four main notions of service: RTC, variable capacity nodes, strict service and minimum service, the question of their relationship naturally arises.

When looking deeply at definition, it also appears that the notion of strict service curve can have two interpretations, one weak and one strict one.

It has been shown in [27, 30] that these notions forms a strict hierarchy, with equality between notion depending on the kind of curves considered. Some monotony results have been also found for each model, allowing to obtain, in most cases, a canonical version of each service curve.

5.2 Strict priority residual services

When several flows share a network element, one may be interested by the service offered to each one (depending on the service policy and the others flows) which is called the "residual service".

This residual service is often assumed to have a wide-sense increasing curve, restricting the kind of usable curves. This restriction has been lifted in [27], and the links with strict service curve has been made explicit.

5.3 Tight results under blind policy

Network calculus, in general, computes pessimistic bounds. One challenge is to get tight bounds, *i.e.* bounds that could be reached. It is well known that bound are tight for a single server [1, Th 1.4.4]. There also are results for FIFO policy in some specific topologies [31].

[28, 29] describe the first algorithm which computes the maximum end-to-end delay for a given flow, as well as the maximum backlog at a server, for *any* feed-forward network topology under blind multiplexing, with concave arrival curves and convex service curves.

5.4 Complexity problem

The computational complexity of the approach in [28, 29] is high, probably supra-exponential. The intrinsic complexity of computing an exact bound have been studied on a specific topology: it is NP-hard [29] (by equivalence with X3C problem, using only two-slopes piecewise linear functions). This complexity suggests that only approximated methods are suited to analyse large systems.

6 Tool support

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.

6.1 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 result.

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

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

6.2 Design considerations

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), the lower the complexity and the more general the class of curves (needed for research), the higher complexity. In order to solve these and other contradictory requirements between industrial and academic usage, the prototype has been structured into several distinct components (see Figure 2), with several different implementations in some cases.

6.3 Implementation

Java has been chosen as programming language for its lower risk for programming errors and because of its link with the Eclipse framework that will be used for the graphical editor. Furthermore continuous integration with frequent releases is performed in order to get rapid feedback from the academic and industrial partners. At the time of writing, the (min,plus) library and the (min,plus) interpreter are fully operational and validated, the network representation (data model) is done and the network calculus core routines are being developed.

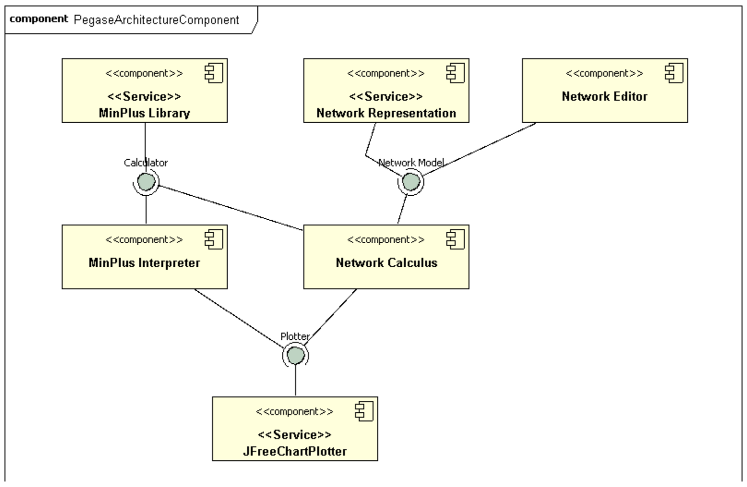

Fig. 2. Components of the prototype (UML notations).

6.4 Tool validation

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 code with the tool SONAR,
– Extensive automated comparison tests with the Network Calculus tool NC-maude [32].

7 Conclusion

The PEGASE project has been submitted in 2009 to the French call ARPEGE from the ANR (National Research Agency) and it has been selected for funding. The project has started in October 2009 for a duration of 36 months. News from the project, and some of its outcomes, can be found on the project WEB page [33].

The first year of the project already produced some theoretical results (section 5) and the first part of the tool: a (min,+) interpreter (section 6)[7]. These preliminary theoretical results suggest that exact temporal verification techniques will not probably be suited to large scale systems such as avionics ones. The complexity threshold from which approximate techniques are required remains to be more precisely identified. One of the main ongoing objective is to come up with sound approximation techniques, whose accuracy ideally could be chosen by the user.

[7] This interpreter is available free of charge for non-commercial use and can be downloaded at [34].

References

1. Le Boudec, J.Y., Thiran, P.: Network Calculus. Volume 2050 of LNCS. Springer Verlag (2001) http://lrcwww.epfl.ch/PS_files/NetCal.htm.
2. Chang, C.S.: Performance Guarantees in communication networks. Telecommunication Networks and Computer Systems. Springer (2000)
3. AEEC: Arinc 664p7-1 aircraft data network, part 7, avionics full-duplex switched ethernet network. Technical report, Airlines Electronic Engineering Committee (september 2009)
4. ECSS: Spacewire – links, nodes, routers and networks. Technical Report ECSS-E-ST-50-12C, European cooperation for space standardization (ECSS), ESA-ESTEC, Requirements & standards division, Noordwijk, The Netherland (31 July 2008)
5. Bengtsson, J., Larsen, K.G., Larsson, F., Pettersson, P., Yi, W.: Uppaal - a tool suite for automatic verification of real-time systems. In: Proceedings of the 4th DIMACS Workshop on Verification and Control of Hybrid Systems. Number 1066 in LNCS, New Brunswick, New Jersey (October 1995) 232–243
6. Hune, T., Romijn, J., Va, F.: Linear parametric model checking of timed automata. Journal of Logic and Algebraic Programming (2001)
7. Tindell, K., Clark, J.: Holistic schedulability analysis for distributed hard real-time systems. Microprocess. Microprogram. **40**(2-3) (1994) 117–134
8. Davis, R.I., Burns, A., Bril, R.J., Lukkien, J.J.: Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. Real-Time Systems **35**(3) (april 2007) 239–272
9. Migge, J.: L'ordonnancement sous contraintes temps réel: un modele á base de trajectoires – Real-time scheduling: a trajectory based model. PhD thesis, Univesity of Nice (1999)
10. Martin, S., Minet, P., George, L.: The trajectory approach for the end-to-end response times with non-preemptive FP/EDF*. In: Proceedings of the Int. Conf. on Software Engineering Research and Applications (SERA'04). Volume 3647 of LNCS., Springer (2004) 229–247
11. Cruz, R.L.: A calculus for network delay, part I: Network elements in isolation. IEEE Transactions on information theory **37**(1) (January 1991) 114–131
12. Cruz, R.L.: A calculus for network delay, part II: Network analysis. IEEE Transactions on information theory **37**(1) (January 1991) 132–141
13. Fidler, M.: A survey of deterministic and stochastic service curve models in the network calculus. IEEE Communications Surveys & Tutorials **12**(1) (2010)
14. Baccelli, F., Cohen, G., G.J., O., Quadrat, J.P.: 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.
15. W., T.K., Burns, A.: Guaranteeing message latencies on controller area network (CAN). In: Proceedings of 1st international CAN conference. (1994) 1–11
16. Schmitt, J., Zdarsky, F., Fidler, M.: Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch... In: Proc. of the 27th IEEE International Conference on Computer Communications (INFOCOM 2008). (2008)
17. Lenzini, L., Mingozzi, E., Stea, G.: Delay bounds for FIFO aggegates: a case study. Computer Communications **28** (2004) 287–299
18. Thiele, L., Chakraborty, S., Naedele, M.: Real-time calculus for scheduling hard real-time systems. In: Proceedings of ISCAS'2000. (2000)
19. Jiang, Y., Liu, Y.: Stochastic Network Calculus. Computer Communication and Networks. Springer Verlag (2009)

20. Schmitt, J.B., Zdarsky, F.A.: The DISCO network calculator - a toolbox for worst case analysis. In: Proceedings of the First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'06), Pisa, Italy, ACM (November 2006)
21. Andrews, M.: Instability of FIFO in session-oriented networks. In: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms (SODA'00), Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2000) 440–447
22. Rizzo, G., Le Boudec, J.Y.: Stability and delay bounds in heterogeneous networks of aggregate schedulers. In: Proc. of the 27th IEEE Conference on Computer Communications (INFOCOM 2008). (13-18 April 2008) 1490–1498
23. Jonsson, B., Perathoner, S., Thiele, L., Yi, W.: Cyclic dependencies in modular performance analysis. In: Proc. of the ACM International Conference on Embedded Software (EMSOFT). (2008)
24. Charara, H., Fraboul, C.: Modelling and simulation of an avionics full duplex switched ethernet. In: Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05), Los Alamitos, CA, USA, IEEE Computer Society (2005) 207–212
25. Frances, F., Fraboul, C., Grieu, J.: Using network calculus to optimize AFDX network. In: Proceeding of the 3thd European congress on Embedded Real Time Software (ERTS06), Toulouse (January 2006)
26. Boyer, M., Fraboul, C.: 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), IEEE industrial Electrony Society (May 21-23 2008) 11–20
27. Bouillard, A., Jouhet, L., Thierry, E.: Service curves in Network Calculus: dos and don'ts. Research Report RR-7094, INRIA (2009)
28. Bouillard, A., Jouhet, L., Thierry, E.: Tight performance bounds in the worst-case analysis of feed-forward networks. In: Proc. of the 19th IEEE International Conference on Computer Communications (IEEE INFOCOM 2010). (14-19 2010) 1 –9
29. Bouillard, A., Jouhet, L., Thierry, E.: Tight performance bounds in the worst-case analysis of feed-forward networks. Research Report RR-7012, INRIA (2009)
30. Bouillard, A., Jouhet, L., Thierry, E.: Comparison of different classes of service curves in network calculus. In: Proc. of the 10th International Workshop on Discrete Event Systems (WODES 2010), Technische Universitt Berlin (August 30 - September 1 2010)
31. Lenzini, L., Martorini, L., Mingozzi, E., Stea, G.: Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree network. Performance Evaluations **63** (2005) 956–987
32. Boyer, M.: NC-maude: maude for computation of worst bounds on real-time (embedded) networks. Technical Report 1/16417, ONERA (2010)
33. Boyer, M.: PEGASE home page. http://sites.onera.fr/pegase (2010)
34. RealTime-at-Work (RTaW): Downloadable software. http://www.realtimeatwork.com/?page_id=1217