

Modeling Distributed Real-Time Systems using Adaptive Petri Nets

Olivier BALDELLON*

Jean-Charles FABRE

Matthieu ROY

CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France

Université de Toulouse ; UPS, INSA, INP, ISAE ; UT1, UTM, LAAS ; F-31077 Toulouse Cedex 4, France

Abstract—Industrial systems, like aircrafts or cars, are based on real-time distributed networks and require high level of robustness, reliability and adaptability. This paper first introduces our approach to implement a distributed monitor of real-time properties, and then introduce a new formalism, *adaptive Petri nets*, that will allow to model such complex, distributed and real-time systems.

Keywords—Distributed Systems Models, Petri Net, Real Time

I. INTRODUCTION

On-line monitoring of complex, distributed and real-time systems is a highly complex task that, to our knowledge, has not yet been fully tackled. On-line monitoring of applications (distributed or not) in a centralized way has been an active research domain [5][6] since a few years now, but these approaches cannot handle the case of distributed and real-time systems, and the following two challenges must be addressed to be able to provide online monitoring in the general case :

- the first challenge concerns *property modeling*, i.e., the ability to formally reason on distributed systems with timing constraints. Such formalism should take into account (i) temporal aspects, (ii) distribution and (iii) evolvability. Moreover, for a formalism to be useable by human users, it should permit to express systems' feature in a hierarchical way, to prevent overwhelming complexity of description.
- the second challenge concerns the transformation of formal properties into a *runtime support*, i.e., the “compilation” of properties in a runnable format. The important points for such a transformation are (i) it should maximize offline work to reduce overhead at runtime, (ii) it must provide a distributed scheme for verification of system wide properties, and (iii) it has to provide the most possible efficient way (both in computation and in network load) to verify a given property.

II. OUR APPROACH

Our approach can be described in three steps represented in Figure 1. Each step builds some tools that will allow to “compile” a model into a distributed monitor. The first step is to introduce a new formalism that will allow to describe complex real-time distributed services. The second step will transform (with three operations) the model to adapt it for the last step, that consists in the deployment and distributed execution of the transformed model.

The introduction of a new formal tool that can be used to describe the system is the main subject of this paper.

This work has been supported by the french national agency (ANR) under contract ANR-BLAN-SIMI10-LS-100618-6-01.

* Corresponding author: olivier.baldellon@laas.fr

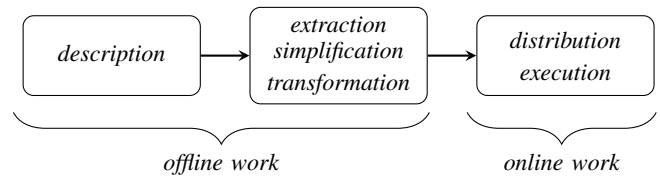


Figure 1. Our approach

This formal approach, called *adaptive Petri net*, is a scalable and hierarchical way to define complex systems that can be modified and adapted.

Then, some transformation work has to be done offline on such a system. The distributed system is a network made of nodes connected together, but because an adaptive Petri net represents a whole system and could be huge, and because we don't want every node to check the whole system, we will extract some parts of this Petri net, a part being a subsystem that a subset of nodes will monitor. This operation is called *extraction*.

The next operation will consist in simplifying the system's parts obtained from the previous operation, for example by abstracting a whole set of places and transitions by a simple place; such an operation is called *simplification*. It could be useful if some nodes are supposed to monitor the global behavior of the system, but do not need to monitor every single operation.

Then we need to transform this piece of adaptive Petri net: as the original adaptive Petri net only describes properties we want to check, we need indeed to transform it to obtain a new model that prevents and/or avoids faults as soon as possible.

The last step will be to deploy and execute the parts to create a distributed real-time monitoring system.

Another approach based on Petri net connected together can be found in [4], [1], but this work do neither consider real-time systems, nor degradations.

III. COMPONENTS AND ARC TIME PETRI NETS

A bottom-up approach will be used to describe our system. The component notion will be the first that will be introduced. A component is the more basic part of the system and is described by arc time Petri net [3]. An arc time Petri net is a classical Petri net where arcs between place and transition are labelled by a time interval (cf. Figure 2).

We suppose that the reader is familiar with the Petri net concept. The main difference between classical Petri nets is that every time a token appears in a place, a new timer starts. Time passes at the same speed in all timers of every token. A token can fire a transition if and only if the value of the timer

is contained in the time interval of the arc. For example, in Figure 2, the first token can fire the transition after a waiting time between 3 and 5 unit. It is easy to see that if the token waits more than 5 it will not be able to fire the transition anymore. In this case the token is said to be *dead*.

To summarize, there are two main rules: *the discrete rule* corresponding to the firings of transitions, and *the continuous rule* corresponding to the passing of time. For a more formal description of Arc time Petri nets, and a comparison with other model time Petri net, the reader can look at [2].

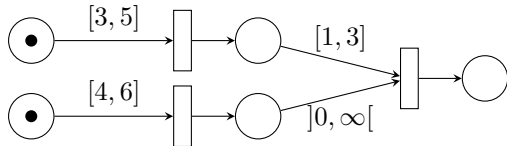


Figure 2. An arc time Petri net

IV. DEGRADABLE COMPONENTS

A degradable component is a sequence of components ($d_0, d_1, d_2, \dots, d_n$) where d_0 is the nominal mode (with no degradation) of the degradable component and the different d_i represent the degraded modes.

We assume that nominal mode and degraded modes have the same interface. The interface of a Petri net is the set of its sources and sinks. The set of sources, the input interface, is the set of place where no transition points to (as i_1 and i_2 in Figure 3). Similarly, the set of sinks, the output interface, is the set of place pointing to no transition (as o_1 in Figure 3).

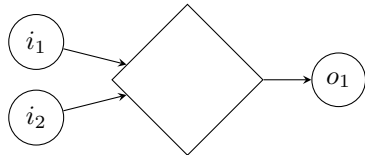


Figure 3. A degradable component (e.g., the example of Figure 2)

V. ADAPTATIVE PETRI NETS

This section shows how a complex system architecture can be described by connecting degradable components. Let us consider the following example: one client and one server. The client can be seen as a set of three components (c_1, c_2, c_3), the server provide three services (a read one r , a write server w_1 and w_2 and a deliver service c). We can first connect with a function f_c the three components c_1, c_2 and c_3 to obtain the client; we can then connect the two operations w_1 and w_2 to obtain the *write* subsystem; the server can be obtained by connecting the three services together with a f_s function to obtain the server. To obtain the whole system we just have to connect the server and the client with an appropriate function.

Due to space limitations, the formal definition of a connection function cannot be described here. The above system can be described by a tree as shown in Figure 4. We call such a tree an *adaptive Petri net*.

In the tree, every branch is a degradable system composed of degradable components. For example if we consider the write subsystem (w_1 and w_2), d_0 represent the default mode (all

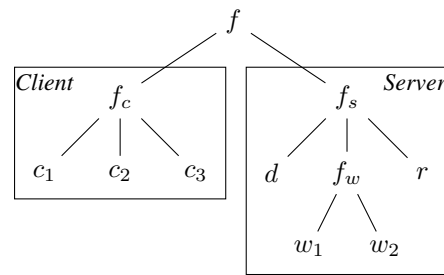


Figure 4. An Adaptive Petri net

information is logged and replicated), d_1 could represent the degraded mode with no replication, d_2 the one with replication but less information logged, d_3 less information logged and no replication and at last d_4 the write service is stopped. In the general case, all d_i are degraded modes of d_0 , but d_{i+1} doesn't have to be a degraded mode of d_i .

In a more formal approach, an adaptive Petri net is a tree whose leaves are degradable components and nodes are connecting functions. This approach allows us to associate a degradable system to each subtree.

CONCLUSION

This paper described a formal approach based on trees and Petri nets to model real-time embedded systems. Our current work include the definition of a clear semantics for such a model. Intuitively, we need to explain (1) how such a model should be executed, (2) when and how it has to be degraded and (3) when and how it must recover, in order to provide a complete solution for resilient systems.

Degradations rules are not defined by the model but by the designers. Degradations rules must reflect priorities between components — which one is the less important and can be degraded to preserve a more important one — and adaptive capabilities of a system: if one component has to be changed, which part of the system do we have to degrade to keep it in a consistent state?

REFERENCES

- [1] A. Benveniste, S. Haar, E. Fabre, and C. Jard. Distributed monitoring of concurrent and asynchronous systems. *CONCUR 2003-Concurrency Theory*, pages 1–26, 2003.
- [2] M. Boyer and O. H. Roux. On the compared expressiveness of arc, place and transition time Petri nets. *Fundamenta Informaticae*, 88(3):225–249, 2008.
- [3] H.M. Hanisch. Analysis of place/transition nets with timed arcs and its application to batch process control. *Application and Theory of Petri Nets 1993*, pages 282–299, 1993.
- [4] A. Madalinski and E. Fabre. Modular construction of finite and complete prefixes of Petri net unfoldings. *Fundamenta Informaticae*, 95(1):219–244, 2009.
- [5] T. Robert, J.C. Fabre, and M. Roy. On-line monitoring of real time applications for early error detection. In *14th IEEE Pacific Rim International Symposium on Dependable Computing*, pages 24–31. IEEE, 2008.
- [6] W. Zhou, O. Sokolsky, B. T. Loo, and I. Lee. DMaC: Distributed Monitoring and Checking. *Lecture Notes in Computer Science*, 5779:184, 2009.



Olivier Baldellon Alumnus of the Rennes extension of the *École Normale Supérieure de Cachan* in the computer science department owns a MSc from University of Rennes 1/IRISA, where he worked on distributed calculability and consensus with Michel Raynal. He is currently doing a PhD supervised by Matthieu Roy and Jean-Charles Fabre on formal tools to implement a distributed real-time monitoring system.