DISTA
Università del Piemonte Orientale "A. Avogadro"
Corso Borsalino 54, 15100 Alessandria

PROGETTO MIUR - ISIDE

Stochastic modeling, analysis techniques and tools for dependable reactive systems

TECHNICAL REPORT - TR-INF-2002-10-05-UNIPMN
February 2002

Modeling the watchdog mechanism of the CESI application through GSPN and Fluid Petri nets

Authors: Simona Bernardi, Marco Gribaudo and Andrea Bobbio

Contents

Modeling the watchdog mechanism of the CESI application through GSPN and Fluid Petri nets

Simona Bernardi[1], Marco Gribaudo[1] and Andrea Bobbio[2]

[1]Dipartimento di Informatica, Università di Torino, 10149 Torino, Italy
[2]DISTA, Università del Piemonte Orientale, 15100 Alessandria, Italy.

1 Introduction

This report discusses the modeling and analysis of the watchdog (WD) mechanism implemented as a part of the TIRAN framework [1]. The functioning, the properties and the main characteristics of the WD have been extensively discussed in [2]. In particular, in [2] a GSPN representation of the WD mechanism has been provided. Due to the nature of the chosen modeling language (the GSPN), in order to provide numerical results, all the timed activities must be assumed exponentially distributed. This assumption is particularly inconsistent when applied to the time-out duration that is typically a deterministic value.

The present report intends to analyze the WD mechanism, when the time-out duration is assumed as deterministic. Various attempts have been documented in the literature to combine deterministic durations with exponential durations in the same PN model, or, more generally, to include non-exponentially distributed transitions [3].

In the present report, we have considered a PN derived model called *Fluid Petri Net (FPN)*. FPN's are an extension of Petri nets able to model systems with the coexistence of discrete and continuous variables [4, 5, 6]. The main characteristics of FPN is that the primitives (places, transitions and arcs) are partitioned in two groups: discrete primitives that handle discrete tokens (as in standard Petri nets) and continuous (or fluid) primitives that handle continuous quantities (referred to as fluid). Hence, in the

single formalism, both discrete and continuous variables can be accommodated and their mutual interaction represented. Furthermore, it has been shown in [7], that the FPN formalism can account for the known non-Markovian PN models.

2 The Watchdog mechanism and the GSPN model

The watchdog mechanism (WD) of the TIRAN framework has been extensively discussed in [2] where the need for a compositional approach to large models is emphasized and it is shown how composition can be achieved.

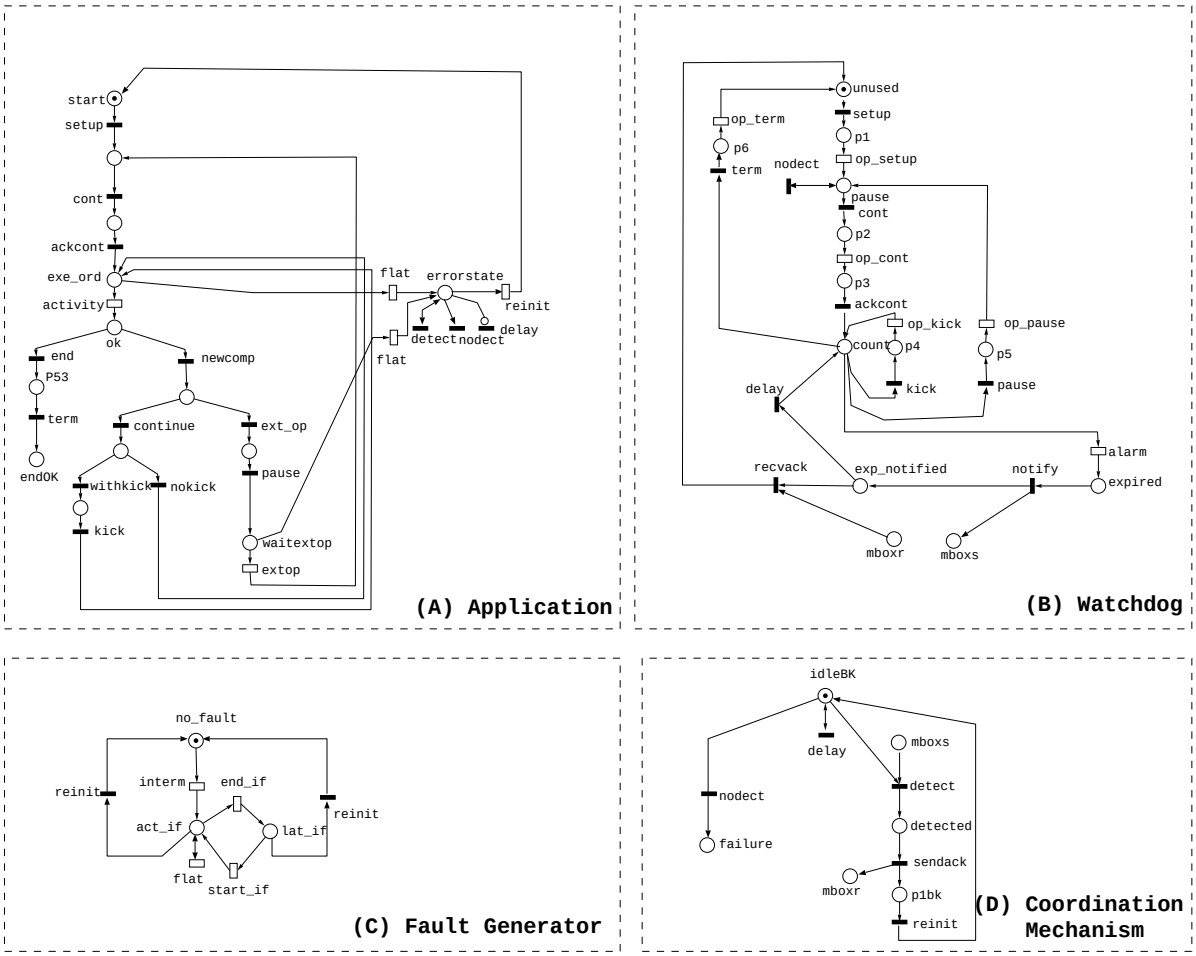

Figure 1: Basic GSPN sub-models used to obtain final analysable GSPN models.

We will consider several GSPN models for the comparative analysis with the corresponding FPN models. These GSPN models are derived by the composition of different GSPN components; the four GSPN sub-models depicted in Figure 1 are the basic GSPN components used to construct the complete GSPN models suitable for the transient

analysis and they are similar to the ones described in [2] with slight changes.

Formally, let $\mathcal{LS}_{ap}, \mathcal{LS}_{wd}, \mathcal{LS}_{ft}$ and $\mathcal{LS}_{cm}$ be the GSPN labeled sub-models (A,B,C) and (D), respectively, depicted in Figure 1 the resulting composed GSPN model $\mathcal{LS}$ is defined as follows:

$$\mathcal{LS} = \mathcal{LS}_{ap} \underset{L^T, L^P}{||} \mathcal{LS}_{wd} \underset{L^T, L^P}{||} \mathcal{LS}_{ft} \underset{L^T, L^P}{||} \mathcal{LS}_{cm} \tag{1}$$

where || is the superposition operator defined in [10] and the superposition is carried out over the set of transition labels:

$$L^T = \{setup, cont, ackcont, term, kick, pause, nodect, delay, detect, flat, reinit\}$$

and over the set of place labels:

$$L^P = \{mboxs, mboxr\}.$$

For abuse of notation, labels and names of transitions/places are the same.

The GSPN components represent respectively:

- The application (see Figure 1(A))
  The behavior of the application in absence of faults consists of 1) starting the WD, represented by the transition setup, 2) setting the timer of the WD, represented the by transition cont and receiving the corresponding acknowledgement from the WD, and 3) executing its own activity represented by the timed transition activity. After finished its current activity, the application can either terminate its execution, and terminate the WD as well, or continue the execution of a similar activity. This choice is modeled by the two conflicting immediate transitions end and newcomp, respectively. In case of continuation, the application can either pause the timer and restart it after a duration represented by the timed activity extop (this choice is represented by the firing of the immediate transition ext_op), or it can choose between two options: simply leaving the timer to countdown to zero or resetting it. This last choice is modeled by the two conflicting immediate transitions nokick and withkick.

  When the application is affected by a fault, after an error latency period that is modeled by the duration of the timed transitions flat, it moves to a state of error (place errorstate becomes marked). In case of error, the application can recover after a duration represented by the timed transition reinit: recovery action is carried out by the coordination mechanism after the error has been detected by the WD. A non detected error brings the application to a failure. Moreover, the presence of an error in the application may be due to a false alarm caused by a delay of the activity execution.

  These different situations - i.e., recovery action after error detection, non detected error and false alarms - are captured by the immediate transitions detect, nodect

and delay. In particular, transition detect tests the state of error of the application and, when the two sub-models representing the application and the coordination mechanism are composed, it is synchronized with the homonymous transition belonging to the GSPN sub-model of Figure 1(D). Transition delay tests the absence of an error in the application and its firing represents the detection of a false alarm (it is synchronized with the homonynous transitions belonging to the GSPN sub-models of Figure 1(B) and (D)). Finally, transition nodect is synchronized with the homonymous transition belonging to the GSPN sub-model of Figure 1(D).

- The watchdog (see Figure 1(B))
  The time-out duration is modeled by the timed transition alarm. The application and the WD submodels interact with each other through the following actions (that in the model are represented by the homonymous immediate transitions): 1) the setup action that activates the WD; 2) the cont action that sets the WD timer to its initial value; 3) the ackcont action, performed by the WD, that aknowledges the application that the setup operation has been carried out; 4) the pause action that stops the WD counting-down 5) the kick action that restarts the WD and, finally, the 6) term action that terminates the WD. Timed transitions op_setup, op_cont, op_ackcont, op_kick, op_pause, op_term model the duration of the corresponding actions. In case of time-out expiration the WD provides to notify the event to the coordination mechanism and then waits for the reception of the acknoledgement. Once received the acknoledgement from the coordination mechanism, the WD returns back to its initial state (place unused becomes marked).

- The fault generator (see Figure 1(C))
  With respect to the fault generator model described in [2], we consider here only intermittent faults. The absence of fault is modeled by a token present in place no_fault. The other two places of the GSPN sub-model act_if and lat_if represent the fault being latent or active, respectively. An active fault can provoke an error to occur in the application: the occurrence of the error is represented by the firing of timed transition flat.

  Transitions flat and reinit interact with the homonymous transitions present in the application model.

- The coordination mechanism (see Figure 1(D))
  Finally, the coordination mechanism models how the error recovery mechanism reacts to a time-out expiration. No timed transitions are included in this sub-net.

  The coordination mechanism model interacts with all the previous models; in particular, with the application model through transitions reinit, delay, nodect and detect; with the fault generator through transition reinit and with the WD through transitions delay, nodect and places mboxs, mboxr.

Fault assumption:

Note that a fault can affect the application either when it is executing its own activity (i,e., the place exe_ord of the application model is marked) or when it is waiting for the termination of an external operation (i.e, the place waitextop of the application model is marked).

As in [2] we assume that the WD and the coordination mechanism, as well as the communication between different components, are tolerant to faults.

Finally, at most a fault can occur in the system at a time ("one fault assumption").

In the GSPN representation of Figure 1, all the timed transitions are forced to be assumed exponential.

3 The FPN Model

Fluid Petri Nets (FPN) are an extension of standard Petri Nets, where, beyond the normal places that contain a discrete number of tokens, new places are added that contain a continuous quantity (fluid). Two main formalisms have been developed in the area of FPN: the Continuous or Hybrid Petri net (HPN) formalism [4], and the Fluid Stochastic Petri net (FSPN) formalism [5, 6]. A complete presentation of FPN is beyond the scope of the present technical report but an extensive discussion of FPN in performance analysis with several examples and case studies can be found in [8].

Discrete places are drawn according to the standard notation and contain a discrete number of tokens that are moved along discrete arcs. Fluid places are drawn by two concentric circles and contain a real variable (the fluid level). The fluid flows along fluid arcs (drawn by a double line to suggest a pipe) according to an instantaneous flow rate. The discrete part of the FPN regulates the flow of the fluid through the continuous part, and the enabling conditions of a transition depend only on the discrete part.

Our FPN formalism contains a new construct called *flush-out arc* [6] (represented at the net level with heavy lines). A flush-out arc is directed from a fluid place to a transition and has the effect of instantaneously empty the place as the transition becomes enabled.

It has been shown in [6], that FPN with the flush-out arc extension can be utilized to model non-exponential stochastic systems and have a greater modeling power with respect to non exponential Petri nets. The presence of non-memory less transitions entails the need of assigning a "memory" policy, as extensively discussed in [9, 7].

In the present case, the fluid construct has been used to represent more correctly, the firing delay of transition alarm (the WD time-out) as a deterministic value. Figure 2 shows how the sub-models of the application and of the WD have been modified in order to obtain the complete analysable FPN models.

The effect of a deterministic time-out can be modeled as follows: transition alarm is

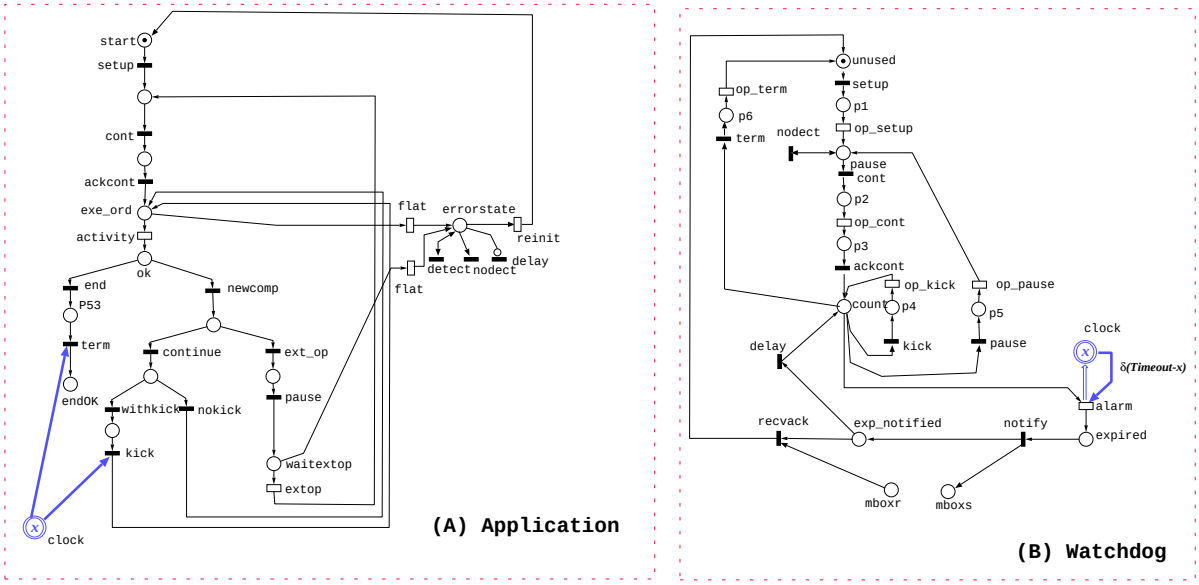

Figure 2: FPN sub-models of the application and of the watchdog.

a timed transition to which a firing rate equal to a Dirac *delta* function is assigned:

$$F_{alarm} = \delta\left(Timeout - x\right)$$

where x is the fluid level in place clock. The fluid arc from transition alarm to place clock has a fluid rate equal to $(r = 1)$, so that fluid level x counts the time in which transition alarm has been enabled. As soon as the fluid level x reaches the $Timeout$ value, transition alarm fires.

The heavy lined arcs emerging from place clock are *flush-out* arcs [6] whose effect is to instantaneously empty the fluid input place clock when enabled. Transition alarm has a mixed *prs/prd* memory policy: the memory is reset due to a kick or a term signal (since those events reset the watchdog), but it is kept when a pause action occurs. This effect is reproduced by the flush-out arcs that connect fluid place clock to transitions clock and term, but not to transition pause.

4 Results

In order to show the effect of using an exponential approximation to a deterministic time-out, various measures have been computed and the results obtained by using GSPN models and FSPN models have been compared. In the following sections we will present

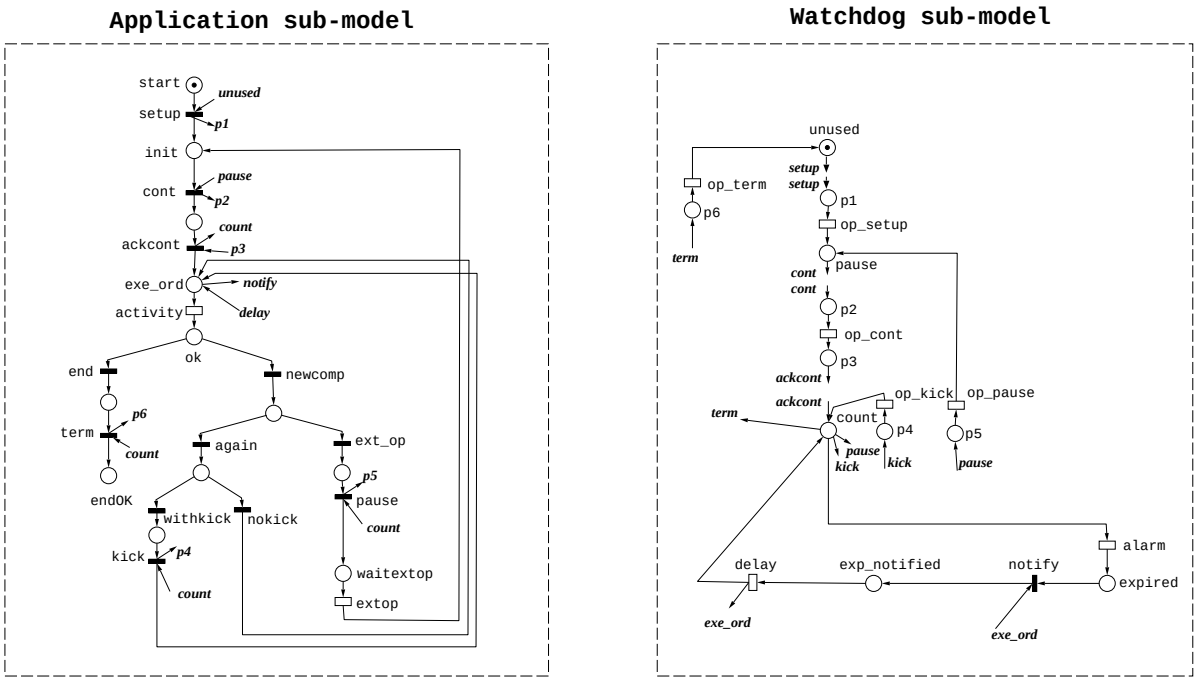

Figure 3: GSPN used to compute the throughput of the alarm.

the complete GSPN models used to compute specific performance indices[1], together with the results obtained in case of usage of GSPN models and in case of FSPN models. The complete FSPN models have been obtained from the corresponding complete GSPN models by modifying the submodels representing the application and the WD as shown in Figure 2. The values assigned to the rate/weight of the transitions of each complete model used for the analysis are listed in tables contained in the appendix A

4.1 Throughput of transition alarm

The throughput of transition alarm has been computed in case of absence of faults.

To obtain the complete GSPN model depicted in Figure 3 we have considered then only the interaction setup, cont, ackcont, term, pause, kick and delay between the application component and the WD component shown in Figure 1(A,B) and from the composed model we have removed the following net objects: transitions flat, detect, nodect, delay, reinit and place errorstate belonging to the original sub-model of the application; transitions nodect, recvack and places mboxs, mboxr beloging to the original sub-model of the WD. Moreover, to penalize the application completion time which is another performance measure that we computed by using this model, we changed the transition

[1]For readability purpose, broken arcs of superposed net objects have been introduced.

delay of the WD from immediate to timed and we added the arcs $(notify, exe\_ord)$ and $(delay, exe\_ord)$.

The complete FSPN model corresponding to the GSPN model of Figure 3 has been obtained by using the FSPN translator `net2fspn`, that transforms the `.net` description files of the *GreatSPN* tool into the `.fspn` input files for the FSPN analysis tool, and by adding the fluid construct to the resulted translated net according to the Figure 2. The values assigned to the rate/weight of the transitions of the model are shown in column (A) of Table 1 of the appendix A.

We have measured the transient throughput of transition alarm. Since no-faults are present this measure represents the expected instantaneous number of false alarms as a function of the time, i.e. the expected number of times the WD erroneously fires at time t because of the presence of a correct but too long activity. The results are presented in Figure 4 for the GSPN (solid line) and for the FPN (dotted line).

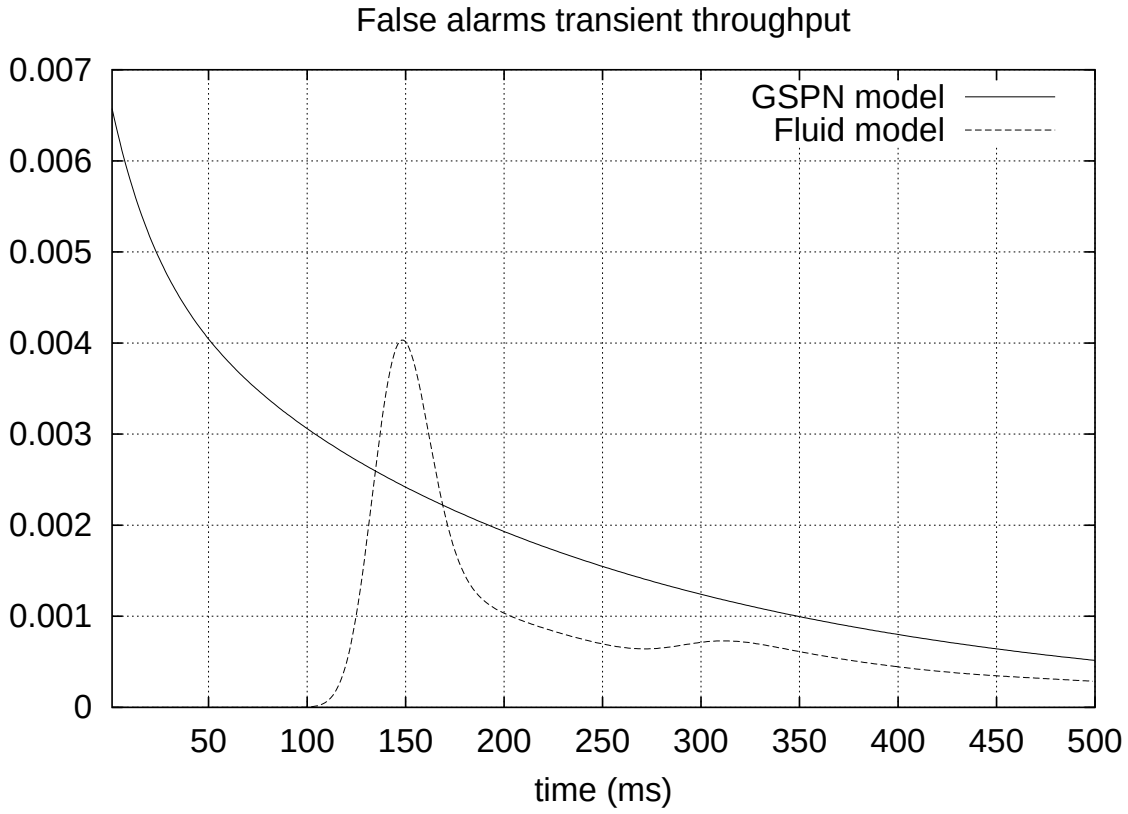

Figure 4: Throughput of transition alarm.

As you can see, the deterministic version some peaks at the integer multiple of the watchdog interval. The first peak should indeed be a discontinuity, and its spread is

caused by the discretization errors. It corresponds to the first firing of the alarm in case of the first false allarm. In the exponential version, the alarm can also fire before 150 ms, which is shown by the decreasing curve.

4.2 Application reliability

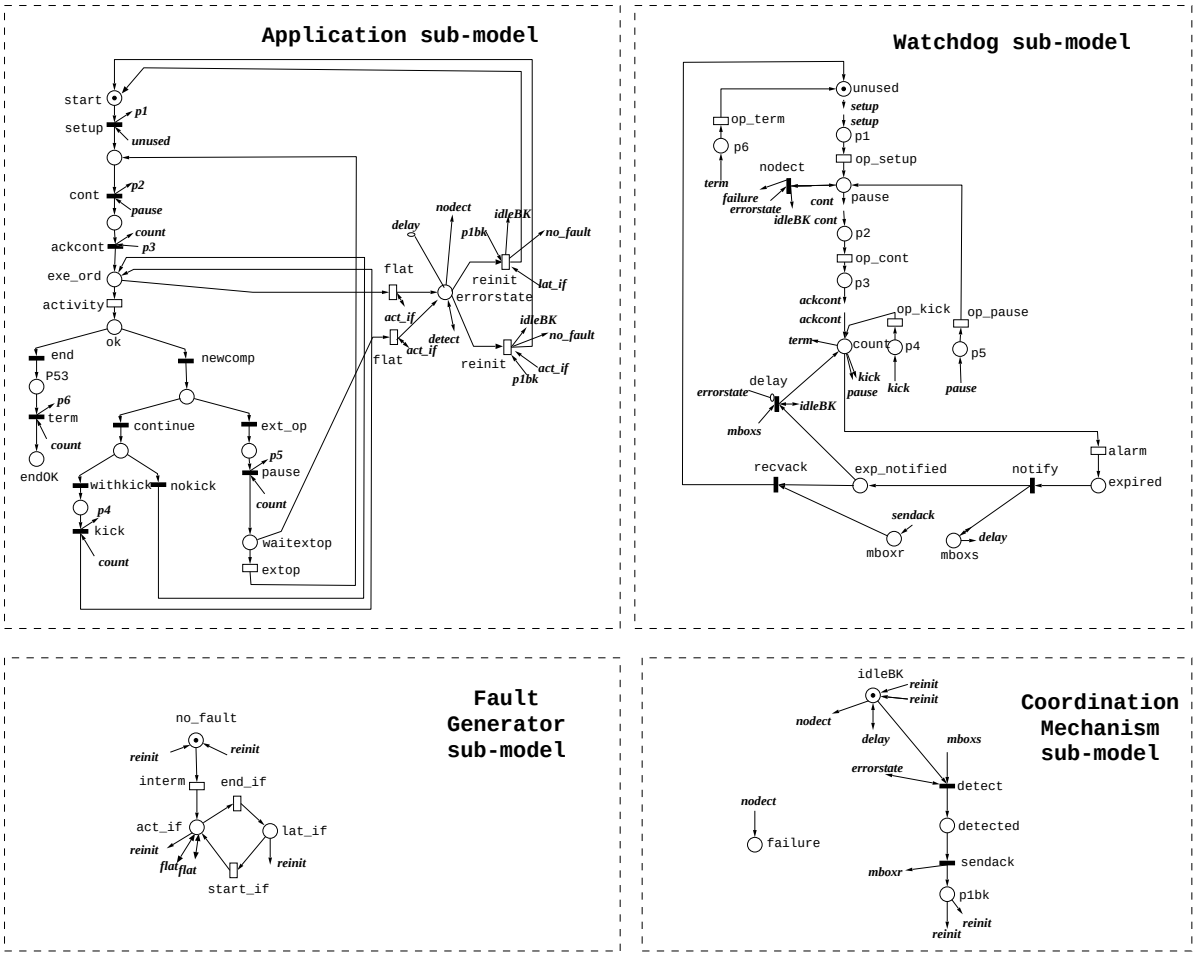

Figure 5: GSPN used to evaluate the application reliability.

In presence of faults, we have evaluated the application reliability. The complete GSPN model depicted in Figure 5 corresponds to the GSPN model $\mathcal{LS}$ given by the formula 1 and it results from the composition of all the four components of Figure 1(A,B,C,D) by considering all the possible interactions among them.

The values assigned to the rate/weight of transitions of the complete models are shown in column (B) of Table 1 of the appendix A.

In the complete models (both GSPN and FSPN), the application *Reliability*, is measured by computing the probability that place labeled failure is not marked at time t:

$$R(t) = 1 - Pr\{\#failure(t) > 0\}$$

The results are presented in Figure 6 for the GSPN (solid line) and for the FSPN (dotted line).

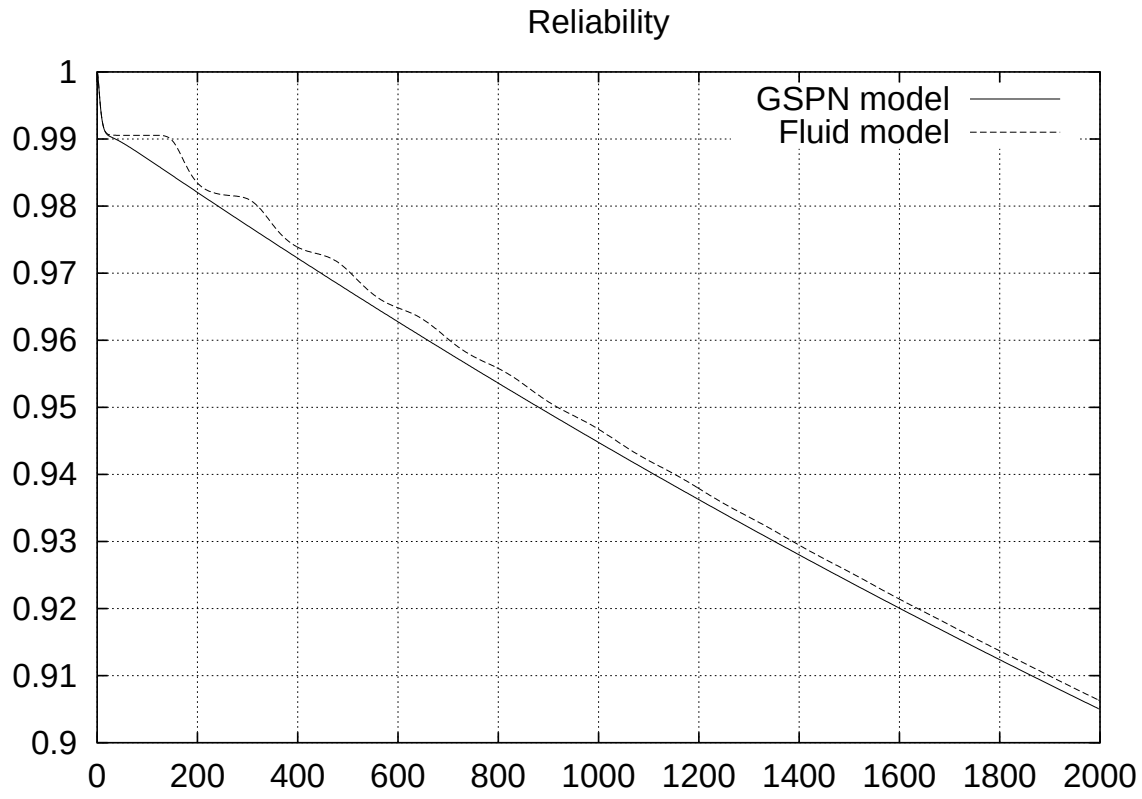

Figure 6: Application reliability.

The Figure shows that the exponential model underestimate the reliability of the model. The "ladder-like" shape of the deterministic version, shows the effect of the watchdog on the reliability of the system. This represents the fact that some of the system failures may be corrected by the watchdog as soon as it fires.

4.3 Application completion time

In presence of faults and under the assumption that all the fault occurrences are tolerated by the application, due to the error detection mechanism and to the coordination mechanism, we have evaluated the application completion time distribution.

The complete GSPN used to compute this performance measure is derived from the complete GSPN model of Figure 5 by removing the subnet of the model that "becomes active" when the application executes the pause action and waits for the external operation to be terminated. The resulting GSPN model obtained from this *cutting* operation is depicted in Figure 7. The values assigned to the rates/weights of the transitions of the model are shown in column (C) of Table 1 of the appendix A.

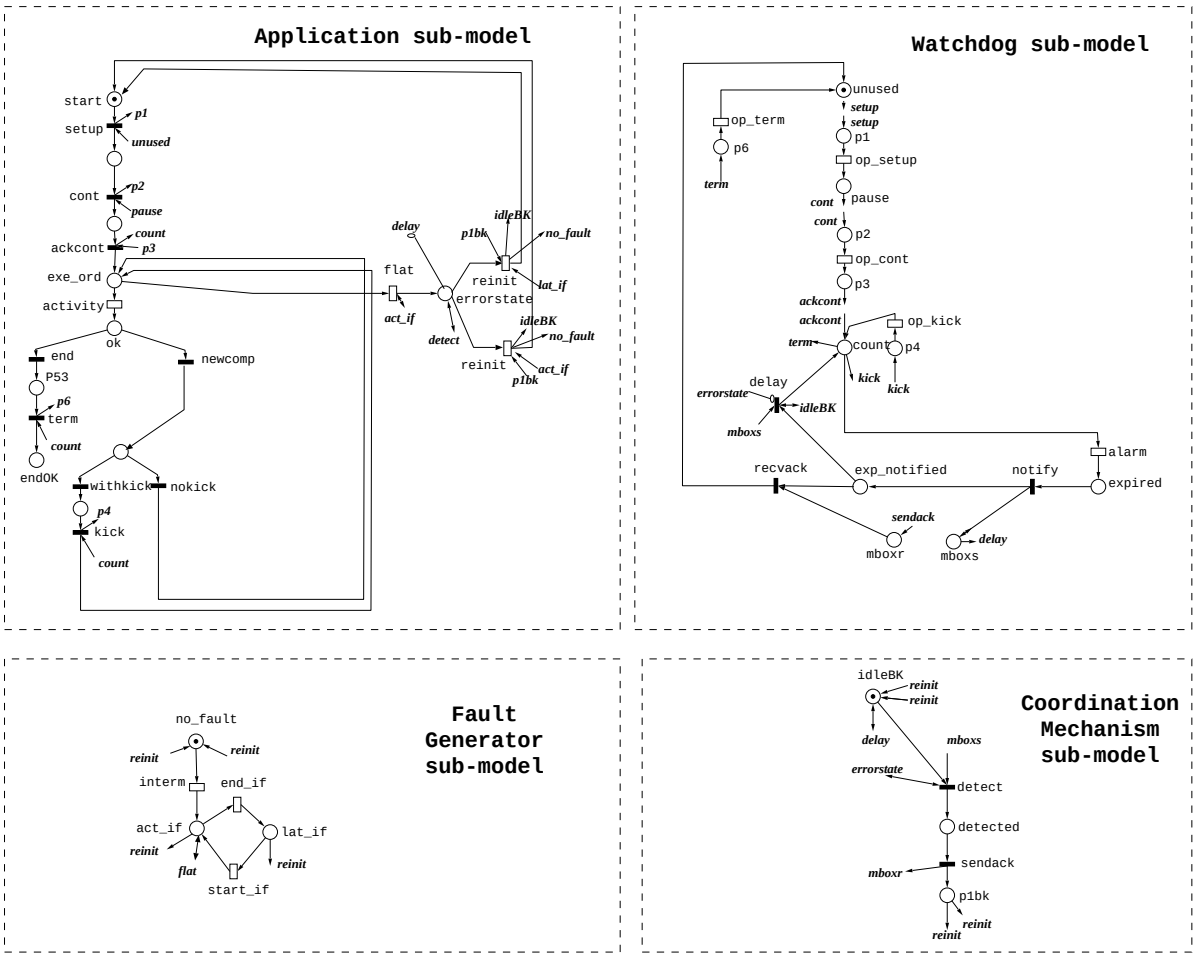

Figure 7: GSPN used to compute the application completion time.

The completion time distribution is evaluated by computing the following probability:

$$C(t) = Pr\{\#endOK(t) > 0\}$$

where #endOK is the place that, when marked, indicates the termination of the application. In case of fault, the application is re-initialized after an exponentially distributed amount of time.

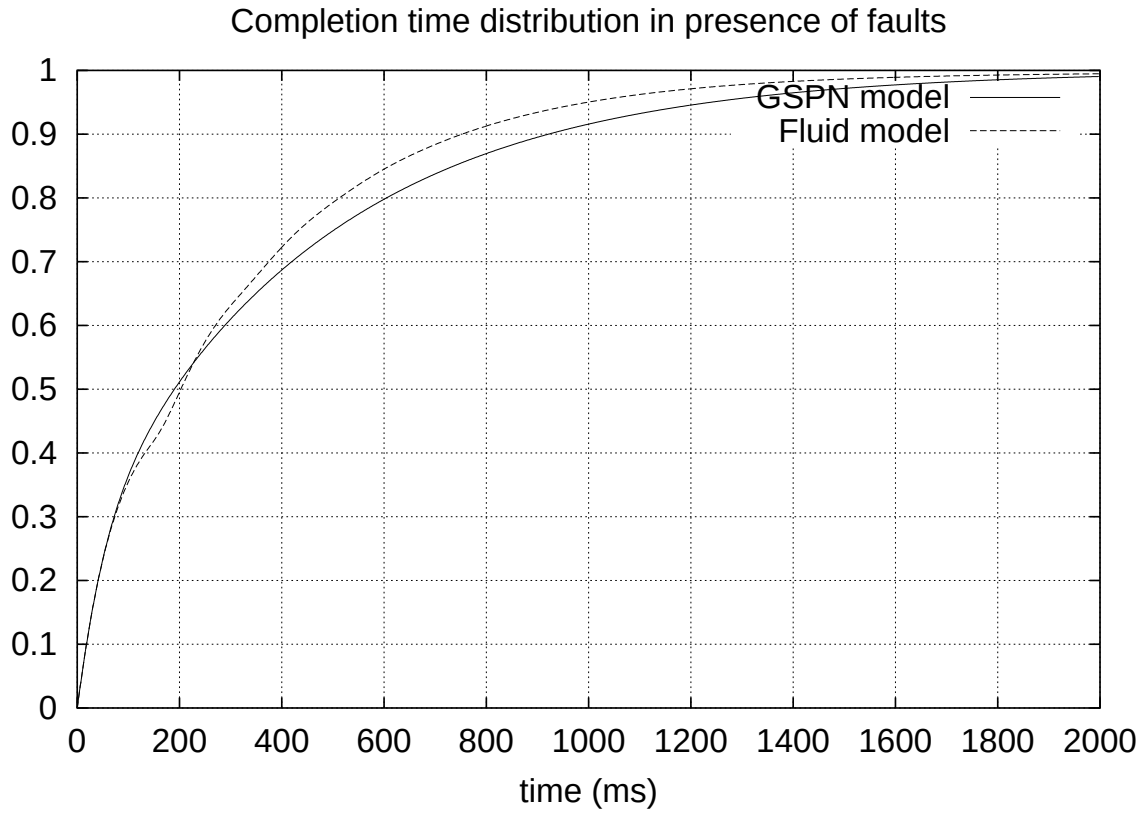

Figure 8: Application completion time.

The figure shows that the completion time for shorter jobs is somehow penalized by the deterministic behavior of the watchdog (the flexus around $t = 150$), but that it can provide better performance for longer jobs, since it ensures that the fault is detected into at most 150 ms.

References

[1] O. Botti, V. De Florio, G. Deconinck, R. Lauwereins, F. Cassinari, S. Donatelli, A. Bobbio, A. Klein, H. Kufner, E. Thurner, , and E. Verhulst. The TIRAN approach to reusing software implemented fault tolerance. In *8th Euromicro Workshop on Parallel and Distributed Processing (PDP2000)*, pages 325–332. IEEE Computer Society, 2000.

[2] S. Bernardi and S. Donatelli. Performance validation of fault-tolerance software: A compositional approach. In *Int. Conference on Dependable Systems and Networks - DSN2001*. IEEE Computer Society, 2001.

[3] A. Bobbio, A. Puliafito, M. Telek, and K. Trivedi. Recent developments in non-Markovian stochastic Petri nets. *Journal of Systems Circuits and Computers*, 8(1):119–158, Feb 1998.

[4] H. Alla and R. David. Continuous and hybrid Petri nets. *Journal of Systems Circuits and Computers*, 8(1):159–188, Feb 1998.

[5] G. Horton, V. Kulkarni, D. Nicol, and K. Trivedi. Fluid stochastic Petri nets: Theory, application and solution techniques. *European Journal of Operational Research*, 105(1):184–201, 1998.

[6] M. Gribaudo, M. Sereno, A. Horváth, and A. Bobbio. Fluid stochastic Petri nets augmented with flush-out arcs: Modelling and analysis. *Discrete Event Dynamic Systems*, 11 (1/2):97–117, January 2001.

[7] M. Gribaudo, M. Sereno, and A. Bobbio. Fluid Stochastic Petri Nets: An extended formalism to include non-markovian models. In *8-th International Conference on Petri Nets and Performance Models - PNPM99*, pages 71–82. IEEE Computer Society, 1999.

[8] M. Gribaudo. Hybrid formalism for performance evaluation: Theory and applications. Technical report, Phd Thesis, Dipartimento di Informatica, Università di Torino, 2001.

[9] A. Bobbio, A. Puliafito, and M. Telek. A modeling framework to implement preemption policies in non-Markovian SPN. *IEEE Transactions Software Engineering*, 26:36–54, 2000.

[10] S. Bernardi, S. Donatelli, and J. Merseguer. From UML sequence diagrams and statecharts to analysable Petri ne t models. In *Proceedings of the Third International Workshop on Software and Performance, (WOSP2002)*, pages 35–45. ACM Press, July 2002.

A Values assigned to the rate/weight of transitions

Transition	Description	Rate (1/ms.)/ Weight		
		(A)	(B)	(C)
activity	execution of normal operations	0.02	0.02	0.02
extop	waiting for external operations	0.02	0.02	-
alarm	WD timeout	1/150	1/150	1/150
op_setup	operation of setup of the WD process	20	20	20
op_cont	operation of setting of the WD timeout	20	20	20
op_kick	operation of kicking the WD alarm	20	20	20
op_term	operation of termination of the WD	20	20	20
op_pause	operation of pausing the WD	20	20	-
delay	delay due to a false alarm	0.04	1	1
flat	fault latency	-	20	0.2
interm	occurrence of an intermittent fault	-	0.4	1/15
start_if	changing the state of the interm. fault from not active to active	-	0.4	0.4
end_if	changing the state of the interm. fault from active to not active	-	10	10
reinit	recovery of the application from a failure	-	0.04	0.04
end	choice of termination of the execution of the application	0.3	0.3	0.3
newcomp	choice of continuation of the execution of the application	0.7	0.7	0.7
ext_op	choice of performing an external activity	0.2	0.2	-
again	choice of repetition of the normal activity	0.8	0.8	-
nokick	choice of not kicking the WD	0.5	0.5	0.5
withkick	choice of kicking the WD	0.5	0.5	0.5

Table 1: Input values for the models used in the analysis: (A) throughput of transition "alarm"; (B) application reliability; (C) application completion time.