Extending GSPNs for the modelling, analysis and performance evaluation of dynamic systems

Samir Tigane, Laid Kahloul* and Samir Bourekkache
LINFI Laboratory,
Computer Science Department,
Biskra University,
Biskra, Algeria
Email: tigane.samir@gmail.com
Email: kahloul2006@yahoo.fr
Email: bourekkache_samir@yahoo.com
*Corresponding author

Souheib Baarir
LIP6 Laboratory,
Computer Science Department,
Paris Ouest Nanterre la Defense University,
Paris, France
Email: souheib.baarir@lip6.fr

Abstract: Cloud-based systems are open systems where resources are theoretically non-limited and the number of connected users/devices varies all the time. These systems are close to dynamic systems where the structure changes at runtime. The objective of this paper is to present a new formalism ‘extended generalised stochastic Petri nets (EGSPNs)’ dedicated to model, analyse and evaluate the performance of reconfigurable systems. In its basic form, GSPNs extend Petri nets by introducing stochastic time. In this paper, GSPNs are enriched with a set of mechanisms allowing the reconfigurability of their structure at runtime. The new formalism ‘EGSPNs’ is applied to model, analyse and evaluate the performance of a case study.

Keywords: Petri nets; generalised stochastic Petri nets; rewriting systems; extended GSPNs; reconfigurable systems; formal modelling; qualitative verification; performance evaluation; dynamic systems; critical systems.


Biographical notes: Samir Tigane obtained his Engineer diploma in Computer Science (Industrial Informatics) from the Computer Science Department, Biskra University in 2006. He obtained his Magister (Software Engineering) in Computer Science from the University of Constantine in 2010. He is a teacher and a PhD student in the Computer Science Department, in the Biskra University. He is a Researcher in the LINFI Laboratory (Laboratory of Smart Computing). His domain of research includes software engineering, formal
methods and reconfigurable systems. He has published previous research papers, at indexed conferences, in the fields of formal methods, reconfigurable systems and Petri nets.

Laid Kahloul obtained his Engineer diploma in Computer Science (Hardware and Software) from the Computer Science Department, Biskra University in 2001. He obtained his Magister (Artificial Intelligence and Images Processing) in Computer Science from the University of Biskra, in 2004. He obtained his PhD in Computer Science from the same university, in 2012. He is an Associate Professor in the Computer Science Department in Biskra University. He is a Researcher in the LINFI Laboratory. His domain of research includes software engineering, formal methods, reconfigurable systems, wireless sensor networks, security and cloud computing. He has published several papers in indexed international journals and indexed conferences.

Samir Bourekkache obtained his Engineer diploma in Computer Science (Industrial informatics) from the Computer Science Department, Biskra University, in 2006. He obtained his Magister (Artificial Intelligence and Distributed Systems) in Computer Science from the University of Constantine in 2009. He obtained his PhD from the University of Biskra, in 2014. He is an Associate Professor in the Computer Science Department, in the same university. He is a Researcher in the LINFI Laboratory. His domain of research includes software engineering, formal methods, reconfigurable systems and cloud computing. He has published several papers in indexed international journals and indexed conferences.

Souheib Baarir obtained his Engineer diploma in Computer Science (hardware and software) from the Computer Science Department, Biskra University in 2000. He obtained his PhD from the University of Paris 6 (UPMC), in 2007. He is an Associate Professor in the Computer Science Department in the “Paris Ouest Nanterre la Defense” University, in Paris, France. He is a Researcher in the LIP6 Laboratory. His domain of research includes software engineering, formal methods, model transformations and SAT problems. He supervised several theses and he has published several papers in indexed international journals and indexed conferences. He has been awarded in many scientific events.

This paper is a revised and expanded version of a paper entitled ‘Net rewriting system for GSPN: a RMS case study’ presented at Advanced Aspects of Software Engineering (ICAASE), Constantine 2 University, 29–30 October 2016.

1 Introduction

Reconfigurable systems are systems with changeable structures at runtime. This category of systems includes mobile robots systems, mobile cellular networks, mobile and dynamic wireless sensor networks, reconfigurable manufacturing systems (RMSs) and several new technological solutions based on cloud computing and Internet of things. Reconfigurable systems require a hard design task at both hardware and software levels. Their reconfigurability is done at runtime; hence the system must not stop working. The use of formal methods in the design of such systems provides several advantages: high
level specification, simulation, properties verification, performance analysis and automatic code generation for software level.

The use of Petri nets (Murata, 1989) in the design, simulation and verification of reconfigurable systems attracts many researchers. Several works have been developed in this domain (Llorens and Oliver, 2004; De Aguiar et al., 2011; Ehrig and Padberg, 2004; Julia et al., 2008; Kahloul et al., 2016; Li et al., 2009; Meng, 2010; Wu and Zhou, 2011; Zhang and Rodrigues, 2009). Recent approaches use high level (reconfigurable) Petri nets which are intuitive and facilitate the modelling process. However, the analysis techniques of reconfigurable Petri nets are still insufficient and not yet well developed. Besides the analysis shortcoming, the highest level Petri nets that deal with reconfigurability do not consider yet time and stochastic events in the proposed formalisms.

In this paper, we are interested to propose a new Petri nets-based formalism for reconfigurable systems. The new formalism is an extension of generalised stochastic Petri nets (EGSPNs) which can be used in the modelling, verification and performance evaluation of reconfigurable systems. The new formalism EGSPNs has three important advantages:

1. it handles stochastic time in the modelling process
2. it provides the reconfigurability of formalism structure
3. it offers a technique for the performance analysis of studied systems.

The remainder of this paper is organised as follows. Section 2 presents related work. Section 3 provides the necessary definition of basic generalised stochastic Petri nets (GSPNs). Section 4 presents the new proposed formalism EGSPN. Section 5 applies the proposed formalism on a data centre case study. Finally, section 6 concludes the paper.

2 Related work

Stochastic Petri nets (SPNs) and GSPNs are extensions of Petri nets (Marsan et al., 1994) which were widely used in literature to model and evaluate systems with stochastic nature. GSPNs handle both qualitative and quantitative properties of modelled systems. Qualitative properties include boundedness, liveness, fairness, deadlockfree, etc. Quantitative properties include efficiency, throughput, power consumption, machine utilisation, etc.

Al-Ahmari and Li (2016) uses GSPNs for the performance investigation of a flexible manufacturing cell consisting of multi-machines. Under different input variable conditions and rate values, the production rate and the utilisation of system components are computed and compared. A variant of SPN, SPNs with fuzzy parameters is used in Tüysüz and Kahraman (2010) to model and analyse a flexible manufacturing cell. The authors in Postema and Haverkort (2014), Philip and Sharma (2013) and Sachdeva et al. (2009) use a specific type of SPNs called stochastic reward nets, where the firing of a transition is associated to an extra condition called guard. The work in Shojafar et al. (2015) combines an artificial intelligence technique called learning automata (LA) with adaptive SPN where each token has its own LA that allows adapting the token flow on the net in order to define the optimal firing sequence.
Besides the above described formalisms, one can distinguish another set of PN-based formalisms that introduce mechanisms to model runtime reconfigurability in PNs structure to treat system reconfigurability. In this set of formalisms, we find:

1. Graph transformation based formalisms [yielding to reconfigurable object nets (RONs) (Biermann and Modica, 2008) and reconfigurable Petri nets (Prange et al., 2008)].

2. Rewriting systems based formalisms [yielding to Badouel’s reconfigurable Petri nets (Llorens and Oliver, 2004) and improved net rewriting systems (INRSs) (Li et al., 2009)].

Using graph grammars, PNs transformation rules are defined and used in the refinement of manufacturing systems (Ehrig and Padberg, 2004). Indeed, the objective is not to provide an approach for the specification and verification of RMSs, but this work reported in Ehrig and Padberg (2004) motivates the researchers to use this approach in RMSs (Prange et al., 2008; Kahloul et al., 2016). In Kahloul et al. (2016) use RONs to propose an approach for the design, simulation and verification of RMSs. Besides graph transformation based techniques, rewriting systems are used in Badouel’s reconfigurable Petri nets (Llorens and Oliver, 2004) which are extended by Li et al. (2009) in INRSs formalism. To handle stochastic events in INRSs, Tigane et al. (2017, 2016) propose an extension for INRSs. However, these extensions focus on the qualitative properties without considering the quantitative ones.

Other formalisms which consider reconfigurability and time in reconfigurable systems are developed in Zhang and Rodrigues (2009), Meng (2010), Wu and Zhou (2011) and Zhang et al. (2013). In Zhang and Rodrigues (2009) use coloured timed Petri nets (CTPNs) in the modelling of RMSs. In order to model reconfigurability, the CTPNs formalism is equipped with reconfigurable transitions, inhibitor arcs and a set of specific places. In Meng (2010) propose coloured timed object oriented nets (CTOONs) where PNs are enriched with oriented object concepts (i.e., derivation, inheritance) and the modularity concept to overcome reconfigurability complexity. In CTOONs the Petri nets are seen as objects in classes and new objects can be derived from other objects; thus reconfiguration process in RMSs is handled using the derivation mechanism in CTOONs.

In Wu and Zhou (2011) intelligent token Petri nets (ITPNs) are proposed where tokens are enriched with time and knowledge. The knowledge enclosed in a token makes some transitions disabled when this token is consumed. In ITPNs new nets can be synthesised from other nets. In Zhang et al. (2013) a formalism called reconfigurable timed net condition/event systems is proposed to deal with the formal modelling and verification of reconfigurable discrete event control systems. In fact, one reconfiguration of the system are supposed to produce a new very similar system. Thus, if the external environment of the unchanged parts is not changed, the repetitive verification of the unchanged parts can be avoided, which reduces the verification cost. However, even these formalisms deal with time and/or reconfigurability, they do not deal with stochastic events.

Although the reconfigurable PNs based approaches offer explicit and intuitive modelling approaches where the structure of the formalism maps the reconfigurable structure of the system, these formalisms stay lacking of analysis techniques and the absence of time and stochastic aspects in the used formalisms.

In this paper we are interested in building an approach to reconfigure GSPNs which considers both following aspects:
Extending GSPNs for the modelling, analysis and performance evaluation

3 Generalised stochastic Petri nets

GSPNs (Marsan et al., 1994) are an extension of Petri nets, where a rate or a weight of firing is associated to each transition. GSPNs represent a generalisation of SPNs by introducing two types of transitions: a set of timed transitions $T_1$ and a set of immediate transitions $T_2$.

When an immediate transition is enabled, it fires in zero time. If there are $n$ enabled immediate transitions $t_1, \ldots, t_n$ with weights, respectively, $w_1, \ldots, w_n$ at a given marking $M$, then the probability that immediate transition $t_i$ fires first is given by:

$$P[t_i \text{ fires first at } M] = \frac{w_i}{\sum_{j=1}^{n} w_j}$$

A timed transition is enabled at a marking $M$ when it meets the conditions as in PNs and in addition $t_i \in T_2$, $t_i$ is not enabled (i.e., there is no enabled immediate transition at marking $M$). Once a timed transition is enabled, it takes a delay to fire which is exponentially distributed. The distribution of random variable $X_i$ (firing delay of transition $t_i$) is given by:

$$F_{X_i}(x) = 1 - e^{\lambda_i x}$$

where $\lambda_i$ is the firing rate.

If there are $n$ enabled timed transitions $t_1, \ldots, t_n$ with rates, respectively, $\lambda_1, \ldots, \lambda_n$ at marking $M$, then the probability that timed transition $t_i$ fires first is given by:

$$P[t_i \text{ fires first at } M] = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$$

The formal definition of GSPNs is given as follows (Marsan et al., 1994):

$$GSPN = (P, T, F, M_0, T_1, T_2, \Lambda)$$

where:

- $(P, T, F, M_0)$ is the underlying PN
- $T_1 \subseteq T$ is the set of timed transitions
- $T_2 \subseteq T$ is the set of immediate transitions
- $T_1 \neq \emptyset$, $T_1 \cap T_2 = \emptyset$, $T_1 \cup T_2 = T$
- $\Lambda : T \rightarrow \mathbb{R}^+$ where $\Lambda_i = \Lambda(t_i)$ is the rate of a negative exponential distribution (resp. weight) associated to timed (resp. immediate) transition $t_i$. 

Qualitative properties: to allow the reconfiguration of a live, bounded and reversible GSPN (LBR GSPN) to another one.

Quantitative properties: by using a new approach that computes the Markov chain describing the dynamic system behaviour. This Markov chain is used to analyse the reconfigurations effects on the system.
The use of GSPNs allows to perform quantitative analysis, since the reachability graph of a given GSPN describes a discrete-time Markov chain (DTMC) (Marsan et al., 1994). DTMC can be defined as a sequence of random variables $X_1, X_2, X_3, \ldots$ with the Markov property (characterised as ‘memorylessness’), that is, the probability of going to the future state depends only on the present state.

To obtain the DTMC from reachability graph of a GSPN, we proceed as follows. The DTMC state space $\{s_0, \ldots, s_n\}$ is the reachability set $\{M_0, \ldots, M_n\}$ of the GSPN and the transition rate (resp. weight) from state $s_i$ (matches marking $M_i$) to state $s_j$ (matches marking $M_j$) is given by $q_{ij} = \Lambda(t)$, the firing rate (resp. weight) of timed (resp. immediate) transition $t$ from $M_i$ to $M_j$.

For further details about how to compute the steady state distribution of a given DTMC, probability distribution of markings, average number of tokens on a place, token probability density, throughput of timed transitions, etc. The reader can refer to Marsan et al. (1994).

4 Extended generalised stochastic Petri nets

EGSPNs combine both formalisms GSPNs presented previously and INRs. The combination of these two formalisms yields to EGSPNs. The next paragraphs present respectively: the basic ideas of INRs, then how to generalise and to apply INRSs to enrich GSPNs with reconfiguration and finally how the new obtained formalism can be used in performance analysis of reconfigurable systems.

4.1 Improved rewriting systems

The INRS based approach presented in Li et al. (2009) is a net rewriting system used to ‘rapidly’ reconfigure basic PNs (representing RMSs). By ‘rapidly’, we mean that there is no need to verify the behavioural properties (i.e., liveness, boundedness and reversibility) of reconfigured PNs, since it is proved that they do not lose these properties after applying a rewriting rule based on the INRS approach. It consists of the substitution of a subnet of live, bounded and reversible (LBR) PN by another subnet, such that each subnet is an instance of one of the net block classes defined in the INRS approach. It was proved that this transformation leads to another LBR PN.

In previous works Tigane et al. (2016, 2017), we have proved that the direct application of this approach on GSPNs does not guarantee that reconfigured GSPNs preserve their behavioural properties. Thus, we had proposed an INRS to deal with reconfigurability in GSPNs.

However, the proposed extension, previously, has two limits:

1. It works only for off-line reconfiguration, since it does not take into account the reconfiguration at run-time (i.e., the state of the reconfigurable system is not considered).

2. No algorithms were proposed for the quantitative properties analysis of the reconfigurable system under investigation which is a key feature.

In this paper, we extend the previous work such that it is used in reconfiguration of an LBR GSPN to another one rapidly at run-time and moreover it considers the quantitative
Extending GSPNs for the modelling, analysis and performance evaluation

properties by providing a new algorithm. The new proposed algorithm allows computing the Markov chain describing the reconfigurable system behaviour in order to study, analyse and evaluate its performance after applying a set of reconfigurations.

4.2 Improved rewriting systems applied to GSPNs: EGSPNs

We define EGSPN as a couple: \( \Psi = (\mathcal{R}, \mathcal{R}_1, \mathcal{R}_2, G^0) \), where:

- \( G^0 = (P, T, F, M_0, T_1, T_2, A) \) is the initial configuration which is an LBR GSPN.
- \( \mathcal{R}_1 \subseteq \mathcal{R} \) is the set of timed rules
- \( \mathcal{R}_2 \subseteq \mathcal{R} \) is the set of immediate rules
- \( \mathcal{R}_1 \cap \mathcal{R}_2 = 0, \mathcal{R}_1 \cup \mathcal{R}_2 = \mathcal{R} \)
- \( \mathcal{R} = \{ r_i = (L_i, R_i, f^i, \tau^i, V_i, AM_i) \mid i = 0, ..., n \} \) is a finite set of rewriting rules where:
  1. \( L_i \) and \( R_i \) (left and right-hand side, respectively, of rule \( r_i \)) are two GSPNs belonging to the net block classes.
  2. \( \tau^i = (I_{L_i}, I_{R_i}, h) \) (resp. \( \tau^i = (O_{L_i}, O_{R_i}, h') \)) is called the input interface relation (resp. output interface relation) of both nets \( L_i \) and \( R_i \) where \( I_{L_i} \subseteq L_i, I_{R_i} \subseteq R_i \) and \( h : I_{R_i} \to I_{L_i} \) (resp. \( O_{L_i} \subseteq L_i, O_{R_i} \subseteq R_i \) and \( h' : O_{R_i} \to O_{L_i} \)).
  3. \( I_{L_i} \cup I_{R_i} \) is a set of nodes belonging to the same type (either timed transitions, immediate transitions or places).
  4. Idem for \( O_{L_i} \cup O_{R_i} \).
  5. \( f_i : L_i \to g \) is an isomorphism, where \( g \) is a subnet of \( G \) and \( G \) is the current EGSPN configuration to be reconfigured.
  6. \( V_i \) is a firing weight (resp. rate of a negative exponential distribution) specifying the priority (reps. firing delay) of immediate (reps. timed) reconfiguration rule \( r_i \).
  7. \( AM_i \) is the activator marking of rule \( r_i \).

A rule \( r = (L, R, f, \tau, V, AM) \) is applicable on a given GSPN \( G \) iff:

- The image of \( f(L) = g \) is a subnet in current GSPN \( G \).
- \( \forall v \in N_G \backslash N_g, \forall w \in N_g, \text{ if } F(v, w) \neq 0 \lor F(w, v) \neq 0 \text{ then } f^{-1}(w) \in L \cup O \) [where \( N_g \) (resp. \( N_G \)) denotes the set of places and transitions in \( G \) (resp. \( g \)).]
- \( \forall v \in N_G \backslash N_g, \forall w \in I_L, \text{ if } F(v, w) \neq 0 \text{ (resp. } F(w, v) \neq 0 \text{) then } \forall y, y' \in I_L, F(v, f(y)) = F(v, f(y')) (\text{resp. } F(y, f(v)) = F(y, f(v'))). \)
- \( \forall v \in N_G \backslash N_g, \forall w \in O_L, \text{ if } F(v, w) \neq 0 \text{ (resp. } F(w, v) \neq 0 \text{) then } \forall y, y' \in O_L, F(v, f(y)) = F(v, f(y')) (\text{resp. } F(y, f(v)) = F(y, f(v'))). \)
- \( M(p) \geq AM(p), \forall p \in P_G, \) where \( M \) is the current marking of GSPN \( G \) (where \( P_G \) denotes the set of places in \( G \)).
If rule \( r \) is timed, the condition \( \forall r' \in \mathcal{R}'_2 \) (resp. \( \forall t \in T_2 \)), \( r' \) (resp. \( t \)) is not applicable (fireable) at \( AM \) must hold, since applying immediate rules (resp. firing immediate transitions) is prioritised on firing timed rules.

An immediate rule is applied in zero time once it is applicable. A timed rule takes time to be applied once it is applicable.

Applying rewriting rule \( r = (L, R, f, \tau, \tau^*, V, AM) \) on GSPN \( G \) leads to new GSPN \( G' \). Let \( F' \) be the flow relations of \( G' \). For all couple of nodes \( v, w \) in \( G' \), \( F'(v, w) \) is defined by the following:

\[
F'(v, w) =
\begin{cases}
F(v, w) & \text{if } v \not\in N_R \land w \not\in N_R \\
F_R(v, w) & \text{if } v \in N_R \land w \not\in N_R \\
F(f(h(w))), w) & \text{if } v \not\in N_R \land w \in I_R \\
F(f(h'(v)), w) & \text{if } v \in O_R \land w \not\in N_R \\
0 & \text{otherwise.}
\end{cases}
\]  

(1)

Marking \( M' \) of new GSPN \( G' \) is given by

\[
M'(p) = \begin{cases}
M(p) & \text{if } p \not\in P_R \\
M_R(p) & \text{if } p \in P_R
\end{cases}
\]  

(2)

As for the net block classes, we keep the same definitions as presented previously in Tigane et al. (2016, 2017).

As an illustrative example, we consider the reconfigurable system depicted in Figure 1. Firstly, its initial configuration is \( C_0 \) shown in Figure 1(a). Once the marking of place \( p_0 \) in \( C_0 \) becomes three, the system is reconfigured to second configuration \( C_1 \) highlighted in Figure 1(f). The reconfigurable system returns to its initial configuration when the marking of place \( p_2 \) in second configuration \( C_1 \) is equal to three.

**Figure 1** Illustrative example, (a) first configuration \( C_0 \) (b) \( L_1 \) (c) \( R_1 \) (d) \( L_2 \) (e) \( R_2 \) (f) second configuration \( C_1 \)

Reconfiguration rule \( r_1 = (L_1, R_1, f_1, \tau_1, \tau_1^*, V_1, AM_1) \) which models the reconfiguration from configuration \( C_0 \) to configuration \( C_1 \) is formulated as follows:

- \( L_1 \) is a net bloc from the class called single transition (Tigane et al., 2016, 2017), shown in Figure 1(b).
Extending GSPNs for the modelling, analysis and performance evaluation

- $R_1$ is a net bloc from the class called two-parallel open machine graph (Tigane et al., 2016, 2017), depicted in Figure 1(c).
- $r_1 = (\{t\}, \{t', t_3\}, h_1)$ where $h_1(t) = t$ and $h_1(t_3) = t$.
- $r_1^* = (\{t\}, \{t', t_3\}, h_1')$ where $h_1'(t) = t$ and $h_1'(t_3) = t$.
- $f_1(t) = t_1$ where $f_1$ is the mapping of left-hand side $L_1$ of rule $r_1$ in GSPN $C_0$, $t$ is in $L_1$ and $t_1$ is in configuration $C_0$.
- $V_1 = 2$ where $V_1$ is the firing rate of rule $r_1$.
- $AM_1(p_0) = (3)$.
- $r_1$ is a timed reconfiguration rule.

Reconfiguration rule $r_2 = (L_2, R_2, f_2, r_2^*, r_2^*, V_2, AM_2)$ which models the reconfiguration from configuration $C_1$ to configuration $C_0$ is given as follows:
- $L_2$ is a net bloc from the class called two-parallel open machine graph (Tigane et al., 2016, 2017), illustrated in Figure 1(d).
- $R_2$ is a net bloc from the class called single transition (Tigane et al., 2016, 2017), highlighted in Figure 1(e).
- $r_2 = (\{t, t'\}, \{t_1\}, h_2)$ where $h_2(t_1) = t$.
- $r_2^* = (\{t, t'\}, \{t_1\}, h_2')$ where $h_2'(t_1) = t$.
- $f_2(t) = t_1'$ and $f_2(t') = t_5$ where $f_2$ is the mapping of left-hand side $L_2$ of rule $r_2$ in GSPN $C_1$, $t, t'$ is in $L_2$ and $t_1', t_5$ is in configuration $C_1$.
- $V_2 = 1$ where $V_2$ is the firing rate of rule $r_2$.
- $AM_2(p_2) = (3)$.
- $r_2$ is a timed reconfiguration rule.

According to the equation (1), the flow relation between place $p_0$ and $t_1'(F'(p_0, t'))$ in configuration $C_1$ after applying rule $r_1$ on reconfiguration $C_0$ is $F'(p_0, t') = F(p_0, f_1(h_1(t_1'))) = F(p_0, t_1) = 1$, where $h_1(t_1') = t$ and $f_1(t) = t_1$ (according to rule $r_1$). The remaining flow relations can be established by the same way.

4.3 Quantitative properties analysis of EGSPNs

In fact, the reconfiguration of a system affects, eventually, its performance, thus it is required to verify how this reconfiguration affects the overall performance. In this subsection, we describe the used method to build reachability marking graph as a Markov chain of a given EGSPN.

Let $\Psi = (\mathcal{R}, \mathcal{R}_1, \mathcal{R}_2, G^0)$, be an EGSPN where $G^0$ is the initial configuration, $\mathcal{R}$ is the set of rules and $\mathcal{R}_1$ (resp. $\mathcal{R}_2$) is the set of timed (resp. immediate) rules.
Let \( s^j_i = (G^i, M^j) \) be the state that is isomorphic to marking \( M^j \) of GSPN \( G^i \) obtained by applying a set of rules on \( G^i \). Let \( A^{IR} (s^j_i) \) (resp. \( F^{IT} (s^j_i) \)) be the set of enabled immediate rules (resp. immediate transitions) in \( G^i \) at \( M^j \). Let \( A^{TR} (s^j_i) \) (resp. \( F^{TT} (s^j_i) \)) be the set of enabled timed rules (resp. timed transitions) in \( G^i \) at \( M^j \). The Markov chain of this EGSPN is computed as follows.

**Step 1** \textit{Initialisation:} create state \( s^0_0 = (G^0, M_0) \). Set the set of states \( \mathcal{S} = \{s^0_0\} \) and the set of edges \( \mathcal{E} = \emptyset \).

**Step 2** \textit{Applying immediate rules:} for each non-visited state \( s^j_i \in \mathcal{S} \) in which an immediate rule can be applied do:

a. Consider \( s^j_i \) as visited.

b. Apply each applicable immediate rule \( r \) on \( s^j_i \) to obtain a new state \( s^j_k = (G^k, M^l) \) (where \( G^k \) is the obtained GSPN after applying \( r \) on \( G^i \) and \( M^l \) is its marking).

c. Add \( s^j_k \) to \( \mathcal{S} \).

d. Create a new arrow labelled by the name of \( r \) from \( s^j_i \) to \( s^j_k \).

e. Add \( (s^j_i, r, s^j_k) \) to \( \mathcal{E} \).

**Step 3** \textit{Firing immediate transitions:} for each non-visited state \( s^j_i = (G^i, M^j) \) in which an immediate transition can be fired do:

a. Consider \( s^j_i \) as visited.

b. Fire each enabled immediate transition \( t \) in \( s^j_i \) to obtain new state \( s^j_l = (G^l, M^l) \).

c. Add \( s^j_l \) to \( \mathcal{S} \).

d. Create a new arrow labelled by the name of \( t \) from \( s^j_i \) to \( s^j_l \).

e. Add \( (s^j_i, t, s^j_l) \) to \( \mathcal{E} \).

**Step 4** \textit{Applying timed rules and firing timed transitions:} for each non-visited state \( s^j_i = (G^i, M^j) \) in which either timed rule \( tr \) can be applied or timed transition \( tt \) can be fired do:

a. Consider \( s^j_i \) as visited;

b. Apply each applicable timed rule \( tr \) on \( s^j_i \) to obtain new state \( s^j_k = (G^k, M^l) \).

c. Add \( s^j_k \) to \( \mathcal{S} \).

d. Create a new arrow labelled by the name of \( tr \) from \( s^j_i \) to \( s^j_k \).
e Add \((s'_j, tr, s'_i)\) to \(\mathcal{E}\).

f Fire each enabled timed transition \(tt\) in \((G', M_j)\) to obtain new state \(s'_i = (G', M_i)\).

g Add \(s'_i\) to \(\mathcal{S}\).

h Create a new arrow labelled by the name of \(tt\) from \(s'_j\) to \(s'_i\).

i Add \((s'_j, tt, s'_i)\) to \(\mathcal{E}\).

Step 5 End: if exists non-visited state in \(\mathcal{S}\) then go to Step 2, otherwise the Markov chain is completely computed.

Take note that if an immediate rule is applicable on state \(s\) then the immediate transitions, the timed rules and the timed transitions are not enabled at state \(s\). Furthermore, if an immediate transition is enabled at state \(s'\) then the timed rules and the timed transitions are not enabled at \(s'\).

The Markov chain illustrated in Figure 2 is obtained by applying the proposed algorithm on the EGSPN shown in Figure 1.

**Figure 2** Markov chain of EGSPN in Figure 1 where the marking in each state is the marking of \(p_0, p_1\) and \(p_2\) respectively, \(\lambda_i\) stands \(\Lambda(t_i)\), \(\lambda'_i\) stands \(\Delta(t'_i)\) and \(s'_j\) stands for marking \(j\) of configuration \(C_i\).
According to the proposed algorithm, the first step consists of creating state $s_0$ that represents initial marking $M_0$ of first configuration $C_0$. At state $s_0$ there are neither applicable immediate rules nor fireable immediate transitions, so we move to the Step 4 in which transition $t_0$ is fired which yields next marking $M_1$ of configuration $C_0$, represented by state $s_1$. Considering the state $s_1$, rule $r_1$ is applicable (since the marking of $p_2$ is three), so EGSPN is reconfigured to configuration $C_1$ (the transition from state $s_1$ to state $s_2$). When EGSPN reaches state $s_2$, rule $r_2$ becomes applicable (since the marking of $p_0$ is three), by which EGSPN is reconfigured to configuration $C_0$ (the transition from state $s_2$ to state $s_0$). Note that, even so the marking of place $p_0$ is three and the left-hand side of rule $r_2$ is located at configuration $C_1$ at state $s_1$, but rule $r_2$ is not applicable since immediate transition $t_2$ is fireable at this state ($s_1$) which has priority and prevents the application of any timed rule.

5 Case study: data centre

In this section, we illustrate the application of the proposed formalism on a data centre case study. Firstly, we give a description of the structure and the behaviour of the data centre. Secondly, we show how to apply a set of rules in order to reconfigure the initial model of the data centre and how to evaluate its performance. Finally, we conclude by a discussion about results.

5.1 Description and modelling

In this case study, we consider a data centre composed of three servers $S_1$, $S_2$, $S_3$ and two buffers:

1. $buf_1$ with capacity of $p$ spaces that receives jobs with high priority.
2. $buf_2$ with capacity of $q$ spaces that receives jobs with normal priority.

Both of buffers are implemented with the policy ‘first in first out’. Server $S_1$ treats jobs with high priority, whereas server $S_2$ is dedicated to jobs with normal priority. In sake of reducing the power consumption, server $S_3$, initially, is standby. It starts working when the number of waiting prioritised (resp. normal) jobs exceeds threshold $S_h$ (resp. $S_n$). The system has three configurations and the switching from configuration to another is conducted according to the number of waiting jobs.

The described data centre, at its first configuration, is modelled by GSPN $C_0$ depicted in Figure 3.

**Figure 3** Configuration $C_0$ where $H < S_h$ and $N < S_n$. 

![Figure 3](image-url)
The description of places and transitions at configuration $C_0$ are given in Table 1. Once the number of waiting jobs with high priority exceeds threshold $S_h$, server $S_1$ is activated which yields the second configuration. After its activation, server $S_1$ joins server $S_1$ in treating prioritised jobs. This configuration is illustrated in Figure 4.

Table 1  Meanings of places and transitions at configuration $C_0$

<table>
<thead>
<tr>
<th>Place name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H_B</td>
<td>The number of tokens, inside this place, models the number of available spaces in buffer $bu_{fh}$</td>
</tr>
<tr>
<td>H_J</td>
<td>The number of tokens models the number of waiting jobs with high priority</td>
</tr>
<tr>
<td>N_B</td>
<td>The number of tokens, inside this place, models the number of available spaces in buffer $bu_{fn}$</td>
</tr>
<tr>
<td>N_J</td>
<td>The number of tokens models the number of waiting jobs with normal priority</td>
</tr>
<tr>
<td>S_1</td>
<td>A token in $S_1$ means that server $S_1$ has begun treating a job</td>
</tr>
<tr>
<td>S_2</td>
<td>A token in $S_2$ means that server $S_2$ has begun treating a job</td>
</tr>
<tr>
<td>S_1'</td>
<td>A token in $S_1'$ means that server $S_1$ has finished treating a job</td>
</tr>
<tr>
<td>S_2'</td>
<td>A token in $S_2'$ means that server $S_2$ has finished treating a job</td>
</tr>
<tr>
<td>S_1_F</td>
<td>A token in $S_1_F$ means that server $S_1$ is idle</td>
</tr>
<tr>
<td>S_2_F</td>
<td>A token in $S_2_F$ means that server $S_2$ is idle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transition name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H_A</td>
<td>Arrival of a job with high priority</td>
</tr>
<tr>
<td>N_A</td>
<td>Arrival of a job with normal priority</td>
</tr>
<tr>
<td>L_1</td>
<td>Server $S_1$ loads a job from buffer $bu_{fh}$</td>
</tr>
<tr>
<td>L_2</td>
<td>Server $S_1$ loads a job from buffer $bu_{fn}$</td>
</tr>
<tr>
<td>S_1_p</td>
<td>Server $S_1$ processes a prioritised job</td>
</tr>
<tr>
<td>S_2_p</td>
<td>Server $S_1$ processes a normal job</td>
</tr>
<tr>
<td>UL_1</td>
<td>Server $S_1$ unloads a finished job</td>
</tr>
<tr>
<td>UL_2</td>
<td>Server $S_2$ unloads a finished job</td>
</tr>
</tbody>
</table>

Figure 4  Configuration $C_1$ where $S_h \leq h$
The system switches to third configuration $C_2$ if the number of waiting jobs with normal priority is bigger than threshold $S_n$ and the number of prioritised jobs is less than $S_h$. In this configuration $C_2$, servers $S_2$ and $S_3$ along together process jobs with normal priority. This configuration is shown in Figure 5.

The description of the new places and transitions in configurations $C_1$ and $C_2$ are given in Table 2.

### Table 2: Meanings of the new places and transitions at configurations $C_1$ and $C_2$

<table>
<thead>
<tr>
<th>Place name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_3</td>
<td>A token in S_3 means that server S_3 has begun treating a job</td>
</tr>
<tr>
<td>S_3'</td>
<td>A token in S_3' means that server S_3 has finished treating a job</td>
</tr>
<tr>
<td>S_3_F</td>
<td>A token in S_3_F means that server S_3 is idle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transition name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L_3</td>
<td>Server S_3 loads a job</td>
</tr>
<tr>
<td>S_3_p</td>
<td>Server S_3 processes a job</td>
</tr>
<tr>
<td>UL_3</td>
<td>Server S_3 unloads a finished job</td>
</tr>
</tbody>
</table>

### 5.2 Rewriting rules of the system reconfigurations

In this subsection, we model the EGSPN based reconfiguration of the described system. First, we reconfigure configuration $C_0$ to get the second configuration. This reconfiguration is illustrated and modelled by GSPN $C_1$ shown in Figure 4. $C_1$ is obtained by applying timed rule $r_1$ on $C_0$ as follows.
Rule \( r_1 \) means to substitute the mapping at \( C_0 \) of its left-hand side depicted in Figure 6(a) by its right-hand side shown in Figure 6(b). The applying of rule \( r_1 \) models the activation of server \( S_3 \). Rule \( r_1 = (L_1, R_1, f_1, \tau_1, V_1, AM_1) \) is given by

- \( L_1 \) and \( R_1 \) are shown in Figure 6
- \( f_1(L_1) = L_1, f_1(S_1) = S_1, f_1(S_1_p) = S_1_p, f_1(S_1') = S_1', f_1(UL_1) = UL_1, f_1(S_1_F) = S_1_F. \)
- \( \tau_1 = (\{L_1\}, \{L_1, L_3\}, h_1) \)
- \( h_1(L_1) = L_1, h_1(L_3) = L_1 \)
- \( \tau_1^* = (\{UL_1\}, \{UL_1, UL_3\}, h_1^*) \)
- \( h_1^*(UL_1) = UL_1, h_1^*(UL_3) = UL_1 \)
- \( V_1 = 2 \)
- \( AM_1 \) allows to apply the rule when the number of the waiting prioritised jobs exceeds \( S_0 \) (i.e., \( AM_1(H_1) = (S_0) \)).

Once buffer \( bu \) is empty, server \( S_3 \) is deactivated. This reconfiguration is obtained by applying immediate rule \( r_2 \) on \( C_1 \), the resulting GSPN is \( C_0 \).

**Figure 7** Left-hand side and right-hand side of \( r_2 \). (a) \( L_2 \) (b) \( R_2 \)

Rule \( r_2 \) means to substitute the mapping at \( C_1 \) of its left-hand side depicted in Figure 7(a) by its right-hand side depicted in Figure 7(b). Rule \( r_2 = (L_2, R_2, f_2, \tau_2, V_2, AM_2) \) is given by

- \( L_2 \) and \( R_2 \) are shown in Figure 7
- \( f_2(L_1) = L_1, f_2(S_1) = S_1, f_2(S_1_p) = S_1_p, f_2(S_1') = S_1', f_2(UL_1) = UL_1, f_2(S_1_F) = S_1_F, f_2(L_3) = L_3, f_2(S_3) = S_3, f_2(S_3_p) = S_3_p, f_2(S_3') = S_3', f_2(UL_3) = UL_3, f_2(S_3_F) = S_3_F. \)
- \( \tau_2 = (\{L_1, L_3\}, \{L_3\}, h_2) \)
- \( h_2(L_1) = L_1 \)
\[ r_3^* = (\{UL_1, UL_3\}, \{UL_1\}, h_3^*) \]
\[ h_3^*(UL_1) = UL_1 \]
\[ V_2 = 1 \]
\[ AM_3 \text{ allows to apply the rule when buffer } bu f_n \text{ is empty (i.e., } AM_3(H_B) = \langle p \rangle \). \]

Aforementioned, when the number of waiting normal jobs exceeds threshold \( S_n \) and server \( S_3 \) is not yet activated (i.e., the number of waiting jobs with high priority is less than threshold \( S_h \)), server \( S_3 \) is activated to join server \( S_2 \) in processing normal jobs. This reconfiguration is modelled by GSPN \( C_2 \) shown in Figure 5. \( C_2 \) is obtained by applying timed rule \( r_3 \) on \( C_0 \) as follows.

**Figure 8** Left-hand side and right-hand side of \( r_3 \), (a) \( L_3 \) (b) \( R_3 \)

Rule \( r_3 \) means to substitute the mapping at \( C_0 \) of its left-hand side depicted in Figure 8(a) by its right-hand side shown in Figure 8(b). Rule \( r_3 = (L_3, R_3, f_3, *r_3, \tau^*_3, V_3, AM_3) \) is given by

- \( L_3 \) and \( R_3 \) are shown in Figure 8
- \( f_3(L_2) = L_2, f_3(S_2) = S_2, f_3(S_2_p) = S_2_p, f_3(S_2') = S_2', f_3(UL_2) = UL_2, f_3(S_2_F) = S_2_F. \)
- \( *r_3 = (\{L_2\}, \{L_2, L_3\}, h_3) \)
- \( h_3(L_2) = L_2, h_3(L_3) = L_2 \)
- \( \tau^*_3 = (\{UL_2\}, \{UL_2, UL_3\}, h_3^*) \)
- \( h_3^*(UL_2) = UL_2, h_3^*(UL_3) = UL_2 \)
- \( V_3 = 2 \)
- \( AM_3 \text{ allows to apply the rule when the number of the waiting normal jobs exceeds } S_n \) and the waiting jobs with high priority is less than threshold \( S_h \) [i.e., \( AM_3(N_J, H_B) = (S_n, p - S_h + 1) ] \)s.

Once buffer \( bu f_n \) is empty, server \( S_3 \) is deactivated. This reconfiguration is obtained by applying immediate rule \( r_4 \) on \( C_2 \), the resulting GSPN is \( C_0 \).
Rule \( r_4 \) means to substitute the mapping at \( C_2 \) of its left-hand side depicted in Figure 9(a) by its right-hand side depicted in Figure 9(b). Rule \( r_4 = (L_4, R_4, f_4, \tau_4^e, \tau_4^x, V_4, AM_4) \) is given by

- \( L_4 \) and \( R_4 \) are shown in Figure 9

- \( f_4(L_2) = L_2, f_4(S_2) = S_2, f_4(S_2_p) = S_2_p, f_4(S_2') = S_2', f_4(UL_2) = UL_2, f_4(UL_2') = UL_2, f_4(L_3) = L_3, f_4(S_3) = S_3, f_4(S_3_p) = S_3_p, f_4(S_3') = S_3', f_4(UL_3) = UL_3, f_4(S_3_F) = S_3_F. \)

- \( \tau_4 = (\{L_2, L_3\}, \{L_2\}, h_4), h_4(L_2) = L_2 \)

- \( \tau_4^e = (\{UL_2, UL_3\}, \{UL_1, UL_2\}, h'_4, (UL_2) = UL_2 \)

- \( V_4 = I_2 \)

- \( AM_4 \) allows to apply the rule when buffer \( bu_{f_h} \) is empty (i.e., \( AM_2(N_B) = (q) \)).

**Figure 9** Left-hand side and right-hand side of \( ss_4, (a) L_4 \) (b) \( R_4 \)

5.3 Performance evaluation of the data centre case study

On the performance evaluation side, we have developed a tool that computes and analyses the Markov chain accordingly to our proposed algorithm. This tool has been used to simulate the case study and to evaluate its performances. We choose to measure the data centre performances under different rates as illustrated in Table 3. In this simulation, we set the capacity of buffer \( bu_{f_h} \) by \( p = 5 \), the capacity of buffer \( bu_{f_s} \) by \( q = 10 \), threshold \( S_h \) by three and threshold \( S_s \) by seven.

**Table 3** Entry of the different evaluations

<table>
<thead>
<tr>
<th>Rates</th>
<th>( \lambda_{h,i} )</th>
<th>( \lambda_{h,i} )</th>
<th>( \lambda_{s,i,p} )</th>
<th>( \lambda_{s,i,p} )</th>
<th>( \lambda_{s,i,p} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Case 2</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Case 3</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Case 4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Given the above input data, simulation and Markov chain computation are performed. The obtained results are shown in Figures 10 and 11.

Figure 10(a) illustrates the machines utilisations under different cases. In the fourth case, low rate of jobs arrivals (either prioritised or normal) and high production rate of both servers $S_1$ and $S_2$ decrease the number of waiting jobs. Hence, the probability that server $S_3$ starts working decreases also. In relation to server $S_2$, the high rate of normal jobs arrivals in Case 3 increases its utilisation. Moreover, the high rate of production for server $S_2$ with the high rate of its utilisation in Case 3 increase its throughput as depicted in Figure 10(b).

Figure 10  
Statistics about the data centre (1), (a) probability that $S_1/S_2/S_3$ is working (b) throughput of $S_1/S_2/S_3$ (see online version for colours)

Figure 11(a) depicts the probabilities that buffer $bu_{fh}$ and buffer $bu_{fn}$ are full. In the second case, the probability that buffer $bu_{fh}$ is full, is the highest among the others. The high prioritised jobs arrival rate and the low rate of production of both servers $S_1$ and $S_3$ increase the number of waiting prioritised jobs.

Figure 11  
Statistics about the data centre (2), (a) probability that $bu_{fh}/bu_{fn}$ is full (b) mean number of waiting prioritised jobs/normal jobs (see online version for colours)
Figure 11(b) highlights the mean number of waiting jobs in both buffers. In Case 1 and 4, the mean number of waiting prioritised jobs (resp. normal jobs) is less than threshold $S_h = 3$ (resp. $S_n = 7$).

6 Conclusions

Reconfigurable systems are characterised by their changeable structures at runtime. Several systems are included in this category. Data centres are reconfigurable systems which can adapt themselves either to new market requirements, to unpredictable damages or to augment their throughput. Data centres are also critical systems where safety and reliability are indispensable, thus the designer of such systems has a high responsibility. The use of Petri nets as a formal framework has demonstrated several advantages in the design of reconfigurable systems, their verification, and validation and performance evaluation. Researchers propose several extensions of Petri nets dedicated to reconfigurable systems. However, to the best of our knowledge there is not yet an extension for GSPNs that deals with reconfigurable systems. This formalism is important due to its expensiveness power allowing the modelling of timed and stochastic systems. In this paper, we have proposed an extension of GSPNs allowing the reconfiguration of GSPNs. This reconfiguration is formulated using the theory of net rewriting systems which was applied formerly to basic Petri nets. The proposed extension defines a set of net block classes which can be used as parts by the designer of a reconfigurable system. The proposed reconfiguration is assumed to preserve the three properties of Petri nets (liveness, boundedness and reversibility). We proposed also a method that computes the Markov chain from the net rewriting system; hence the performance evaluation can be carried out. In a future work, we will focus on enriching the net block classes by new ones.

References


